WINSMART ACADEMY

MOTTO: PERSONALISED TUTORING FOR LEADERS OF TOMORROW

WHATSAPP: 08024665051, 08169595996

Winsmartacademy@gmail.com

CIT392 - COMPUTER LABORATORY II SUMMARY

Q. What do you understand by programming?

DEFINITION OF PROGRAMMING

Programming is the implementation of logic to facilitate specified computing operations and functionality. It occurs in one or more languages, which differ by application, domain and programming model.

Computer programming is the process of designing and building an executable computer program to accomplish a specific computing result or to perform a specific task. Programming involves tasks such as: analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as coding). The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit. The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem. Proficient programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.

Q. What is a program?

A computer program is a collection of instructions that can be executed by a computer to perform a specific task. A computer program is usually written by a computer programmer in a programming language. A program is either a set of written instructions created by a programmer or an executable piece of software.

What is an Algorithm?

A digital computer is a useful tool for solving a great variety of problems. A solution to a problem is called an algorithm; it describes the sequence of steps to be performed for the problem to be solved. An algorithm is a mathematical process to solve a problem using a finite number of steps. An algorithm is expressed in abstract terms.

Q. Define a compiler and an assembler.

i. Assembler: An assembly language program is translated to machine language by a translator called an assembler.

ii. Compiler: A program written in a high-level language is translated to machine language by a translator called a compiler. Compilers translate source code into a program, which you can run at a later time. Although interpreters are easier to work with, most serious programming is done with compilers because compiled code runs much faster. C++ is a compiled language.

Q. Explain how computer can be used to solve problems.

SOLVING PROBLEMS USING COMPUTER

The problems programmers are asked to solve have been changing. Twenty years ago, programs were created to manage large amounts of raw data. The people writing the code and the people using the program were all computer professionals. Today, computers are in use by far more people, and most know very little about how computers and programs work. Computers are tools used by people who are more interested in solving their business problems than in struggling with the computer.

Ironically, to become easier to use for this new audience, programs have become far more sophisticated. Gone are the days when users typed cryptic commands at esoteric prompts, only to see a stream of raw data. Today's programs use sophisticated "user-friendly interfaces" involving multiple windows, menus, dialog boxes, and the myriad of metaphors with which we have become familiar. The programs written to support this new approach are far more complex than those written only ten years ago.

With the development of the Web, computers have entered a new era of market penetration; more people are using computers than ever before and their expectations are very high. In the last few years, programs have become larger and more complex, and the need for object-oriented programming techniques to manage this complexity has become manifest. As programming requirements have changed, both languages and the techniques used for writing programs have evolved.

Q. Explain what you understand by procedural, structural and Object-Oriented programming?

PROCEDURAL, STRUCTURED, AND OBJECT-ORIENTED PROGRAMMING

Until recently, programs were thought of as a series of procedures that acted upon data. A procedure, or function, is a set of specific instructions executed one after the other. The data was quite separate from the procedures, and the trick in programming was to keep track of which functions called which other functions, and what data was changed. To make sense of this potentially confusing situation, structured programming was created.

The principal idea behind structured programming is as simple as the idea of divide and conquer. A computer program can be thought of as consisting of a set of tasks. Any task that is too complex to be described simply would be broken down into a set of smaller component tasks, until the tasks were sufficiently small and self-contained that they were easily

understood. As an example, computing the average salary of every employee of a company is a rather complex task. You can, however, break it down into the following subtasks:

1. Find out what each person earns.

2. Count how many people you have.

3. Total all the salaries.

4. Divide the total by the number of people you have.

Totalling the salaries can be broken down into the following steps:

1. Get each employee's record.

2. Access the salary.

3. Add the salary to the running total.

4. Get the next employee's record.

In turn, obtaining each employee's record can be broken down into the following:

1. Open the file of employees.

2. Go to the correct record.

3. Read the data from disk

Structured programming remains an enormously successful approach for dealing with complex problems. By the late 1980s, however, some of the deficiencies of structured programming had become all too clear.

First, a natural desire is to think of data (employee records, for example) and what you can do with data (sort, edit, and so on) as a single idea. Procedural programming worked against this, separating data structures from functions that manipulated that data.

Second, programmers found themselves constantly reinventing new solutions to old problems. This is often called "reinventing the wheel," which is the opposite of reusability. The idea behind reusability is to build components that have known properties, and then to be able to plug them into your program as you need them. This is modelled after the hardware world—when an engineer needs a new transistor, she usually does not invent a new one; she goes to the big bin of transistors and finds one that works the way she needs it to, or perhaps modifies it. No similar option existed for a software engineer.

Object-oriented programming attempts to respond to these needs, providing techniques for managing enormous complexity, achieving reuse of software components, and coupling data with the tasks that manipulate that data. The essence of object-oriented programming is to treat data and the procedures that act upon the data as a single object—a self-contained entity with an identity and certain characteristics of its own.

Q. List any six Windows XP commands

1. Copy command
2. Copy this file/folder command
3. Delete this file/folder command
4. Format command
5. Make a new folder command
6. Restore command

Q. Write short notes on the Windows XP commands.

WINDOWS XP COMMANDS

Commands in XP can be found in the menu of applications. When menus are clicked on, a dropdown bar will emerge and these command options can be seen and chosen based on the task at hand. They include the following;

1. **Copy command** -Copies the contents of an external storage device. Found in the menu displayed by right-clicking the content to be copied.
2. **Copy this file/folder command** -Copies a selected file/folder to a new location. Found in the File and Folder Tasks section of the My Computer window.
3. **Delete this file/folder command** -Moves a selected file/folder to the Recycle Bin. Found in the File and Folder Tasks section of the My Computer window.
4. **Format command** -Prepares an external storage devices to receive data. Found in the menu displayed by right-clicking the icon of the device.
5. **Make a new folder command** -Creates a new folder. Found in the File and Folder Tasks section of the My Computer window.
6. **Move this file/folder command** -Moves a file/folder to a new location. Found in the File and Folder Tasks section of the My Computer window.
7. **Rename this file/folder command** -Selects the existing file/folder name so that a new name can be typed. Found in the File and Folder Tasks section of the My Computer window.
8. **Restore command** -Restores a file in the Recycle Bin to its previous location.
9. **Search for files/folders command** -Used to find files and folders. Found in the System Tasks section of the My Computer window.

Q. Enumerate three Non-numeric Data Type in C++ and write short notes on the three

Non-numeric Data Type

1. Boolean Type
2. Character Types
3. String Types

- **Boolean Type**

A Boolean (*bool*) can have only two possible values: True and False. A Boolean is used to express the results of logical operations.

```
#include <iostream>
        int main()
{
        bool married = true;
        bool single = false;
        cout << "The value of married is " << married << endl;
        cout << "The value of single is " << single ;
        return 0;
    }
```

- **Character Types**

A variable of type *cha r* can hold a character of the implementation's character set. For example: *char ch* ='a'; A *char* has 8 bits so that it can hold one of 256 different values.

```
#include <iostream>
        int main()
{
        char character1 = 'a';
        cout << "The value of character1 is " << character1;
        return 0;
}
```

- **String Types**

To store more than one character in a variable, you will need to declare a String data type instead. String is not a fundamental data type. In C++, a String is an object, but a String variable is declared just like any other data type. A reference to the C++ 'string' library is the particular library that contains the code that allows us to use the string object. Some C++ compilers will complain that they can't find the string object if you don't include a reference to the library.

```
#include <iostream>
#include <string> // include for C++ standard string class
int main()
{
        string string1 = "John Smiley";
        cout << "The value of string1 " << string1;
        return 0;
}
```

Q. List any 6 independent libraries for input/output operations in Ada

1. Text I/O

2. Direct I/O

3. Sequential I/O

4. Storage I/O

5. Stream I/O

Q. There are 5 independent libraries for input/output operations in Ada. Enumerate them and write short notes on any 4.

I/O (Input / Output) LIBRARY

There are 5 independent libraries for input/output operations in Ada and they are;

1. Text I/O: This is probably the most used Input/Output library. Text I/O provides support for line and page layout but the standard is free form text. The general form is Ada.Text_IO;

2. Direct I/O: This is used for random access files which contain only elements of one specific type. The general form is Ada.Direct_IO;

3. Sequential I/O: This is used for random access files which contain elements of more than one specific type. You have to read and write the elements one after the other. It just the opposite to Direct I/O. The general form is Ada.Sequential_IO;

4. Storage I/O: This allows you to store one element inside a memory buffer. Storage I/O is useful in Concurrent programming where it can be used to move elements from one task to another.

5. Stream I/O: This is the most powerful input/output library in Ada, it allows you to mix objects from different element types in one sequential file. In order to read/write from/to a streameach type provides a 'Read and 'Write attribute as well as an 'Input and 'Output attribute.

Q. Briefly describe a constant in C++ programming language. Illustrate your answer with example(s).

Symbolic Constants

A symbolic constant is a constant that is represented by a name, just as a variable is represented. Unlike a variable, however, after a constant is initialized, its value can't be changed. If your program has one integer variable named students and another named class, you could compute how many students you have, given a known number of classes, if you knew each class consisted of 15 students:

```
students = classes * 15;
```

In this example, 15 is a literal constant. Your code would be easier to read, and easier to maintain, if you substituted a symbolic constant for this value:

```
students = classes * studentsPerClass
```

If you later decided to change the number of students in each class, you could do so where you define the constant studentsPerClass without having to make a change every place you used that value. Two ways exist to declare a symbolic constant in C++. The old, traditional, and now obsolete way is with a preprocessor directive, #define and the const keyword.

Q. Define a Delimiter in Ada and state the different types.

DELIMITERS

A Delimiter is a character or space marking the beginning or end of a data element. In Ada, there are two categories of delimiters and they include the following with examples under them.

i. Single character delimiters

ii. Compound character delimiters

Q. What is the full meaning of COBOL? and state area where it is used.

Common Business-Oriented Language (COBOL)

COBOL is one of the oldest programming languages. Its name is an acronym for **Common Business-Oriented Language**, defining its primary domain in business, finance, and administrative systems for companies and governments. COBOL programs are in use globally in governmental and military agencies and in commercial enterprises, and are running on operating systems such as IBM's z/OS, the POSIX families (Unix/Linux etc.), and Microsoft's Windows as well as ICL's VME operating system and Unisys' OS 2200. For over four decades COBOL has been the dominant programming language in the business computing domain.

Q. State four (4) common features of COBOL

FEATURES OF COBOL

1. Self-modifying code

The original COBOL specification supported self-modifying code via the infamous "ALTER X TO PROCEED TO Y" statement. X and Y are paragraph labels, and any "GOTO X" statements executed after such an ALTER statement have the meaning "GOTO Y" instead. Most compilers still support it, but it should not be used in new programs.

2. COBOL is Maintainable

COBOL has a proven track record for application maintenance, enhancement and production support at the enterprise level. COBOL applications were actually cheaper to fix than applications written in more recent languages. One reason for the maintainability of COBOL programs has been given above - the readability of COBOL code. Another reason is COBOL's rigid hierarchical structure. In COBOL programs all external references, such as to devices, files, command sequences, collating sequences, the currency symbol and the decimal point symbol, are defined in the Environment Division.

When a COBOL program is moved to a new machine, or has new peripheral devices attached, or is required to work in a different country; COBOL programmers know that the

parts of the program that will have to be altered to accommodate these changes will be isolated in the Environment Division. In other programming languages, programmer discipline could have ensured that the references liable to change were restricted to one part of the program but they could just as easily be spread throughout the program. In COBOL programs, programmers have no choice. COBOL's rigid hierarchical structure ensures that these items are restricted to the Environment Division.

3. Simplicity Feature

COBOL is a simple language (no pointers, no user defined functions, no user defined types) with a limited scope of function. It encourages a simple straightforward programming style. Curiously enough though, despite its limitations, COBOL has proven itself to be well suited to its targeted problem domain (business computing). Most COBOL programs operate in a domain where the program complexity lies in the business rules that have to be encoded rather than in the sophistication of the data structures or algorithms required. And in cases where sophisticated algorithms are required COBOL usually meets the need with an appropriate verb such as the SORT and the SEARCH.

4. Non-proprietary (portable)

Feature The COBOL standard does not belong to any particular vendor. The vendor independent ANSI COBOL committee legislates formal, non-vendor-specific syntax and semantic language standards. COBOL has been ported to virtually every hardware platform - from every favour of Windows, to every falser of Unix, to AS/400, VSE, OS/2, DOS, VMS, Unisys, DG, VM, and MVS.

Q. Briefly describe the types of packages in Ada

THE TYPES OF PACKAGES

I. Nested packages: A nested package is a package declared inside a package. Like a normal package, it has a public part and a private part. A package may also be nested inside a subprogram. In fact, packages can be declared in any declarative part, including those of a block.

II. Child packages: In Ada, you can extend the functionality of a unit (package) with so-called children (child packages). All the functionality of the parent is available to a child including all public and private declarations of the parent package are visible to all child packages.

Q. Write a short note on the concept of Shells in Linux

Linux is a multitasking, multiuser operating system, which means that many people can run many different applications on one computer at the same time. This differs from MS-DOS, where only one person can use the system at any one time. Under Linux, to identify yourself to the system, you must log in, which entails entering your login name (the name the system uses to identify you), and entering your password, which is your personal key for logging in

to your account. Because only you know your password, no one else can log in to the system under your user name.

On traditional UNIX systems, the system administrator assigns you a user name and an initial password when you are given an account on the system. However, because in Linux you are the system administrator, you must set up your own account before you can log in. For the following discussions, we'll use the imaginary user name, ``larry."

In addition, each system has a host name assigned to it. It is this host name that gives your machine a name, gives it character and charm. The host name is used to identify individual machines on a network, but even if your machine isn't networked, it should have a host name. For our examples below, the system's host name is ``mousehouse".

Q. Explain the major function of the kernel and shell in linux .

Shell is a program that takes the commands you type and translates them into instructions to the operating system. A shell is just one interface to Linux. As soon as you log in, the system starts the shell, and you can begin entering commands. Here's a quick example. Larry logs in and is waiting at the shell **prompt**. The last line of this text is the shell's prompt, indicating that it's ready to take commands. The Linux kernel is unique and flexible because it is also modular in nature.

Q. List and discuss the parts of Linux

The parts of the Linux operating system will be examined as follows;

1. The Kernel

All operating systems have kernels, built around the architectural metaphor that there must be a central set of instructions to direct device hardware, surrounded by various modular layers of functionality. The Linux kernel is unique and flexible because it is also modular in nature.

Modularity is desirable because it allows developers to shed parts of the kernel they don't need to use. Typically a smaller kernel is a faster kernel, because it isn't running processes it does not need. If a device developer wants a version of Linux to run on a cell phone, she does not need the kernel functionality that deals with disk drives, Ethernet devices, or big monitor screens. She can pull out those pieces (and others), leaving just the optimized kernel to use for the phone.

The kernel of the Window operating system (which few people outside of Microsoft are allowed to look at without paying for the privilege) is a solidly connected piece of code, unable to be easily broken up into pieces. It is difficult (if not impossible) to pare down the Windows kernel to fit on a phone. This modularity is significant to the success of Linux. Modularity also effects stability and security as well. If one piece of the kernel code happens to fail, the rest of the kernel will not crash. Similarly, an illicit attack on one part of the kernel (or the rest of the operating system) might hamper that part of the code, but should not compromise the security of the whole device.

2. The Operating System

Developers need special tools (like the compilers and command lines found in GNU) to write applications that can talk to the kernel. They also need tools and applications to make it easy for outside applications to access the kernel after the application is written and installed.

This collective set of tools, combined with a kernel, is known as the operating system. It is generally the lowest layer of the computer's software that is accessible by the average user. General users get to the operating system when they access the command line.

Linux provides powerful tools with which to write their applications: developer environments, editors, and compilers are designed to take a developer's code and convert it to something that can access the kernel and get tasks done. Like the kernel, the Linux operating system is also modular. Developers can pick and choose the operating tools to provide users and developers with a new flavor of Linux designed to meet specific tasks.

3. The Environments

The windows, menus, and dialog boxes most people think of as part of the operating system are actually separate layers, known as the windowing system and the desktop environment. These layers provide the human-oriented graphical user interface (GUI) that enables users to easily work with applications in the operating system and third-party applications to be installed on the operating system.

In Linux, there a lot of choices for which windowing system and desktop environment can be used, something that Linux allow users to decide. This cannot be done in Windows and it's difficult to do in OS X. Like the operating system and kernel, there are tools and code libraries available that let application developers to more readily work with these environments (e.g., gtk+ for GNOME, Qt for KDE).

4. The Applications

Operating systems have two kinds of applications: those that are essential components of the operating system itself, and those that users will install later. Closed operating systems, like Windows and OS X, will not let users (or developers) pick and choose the essential component applications they can use. Windows developers must use Microsoft's compiler, windowing system, and so on. Linux application developers have a larger set of choices to develop their application. This allows more flexibility to build an application, but it does mean a developer will need to decide which Linux components to use.

5. The Distributions

This is the highest layer of the Linux operating system: the container for all of the aforementioned layers. A distribution's makers have decided which kernel, operating system tools, environments, and applications to include and ship to users.

Distributions are maintained by private individuals and commercial entities. A distribution can be installed using a CD that contains distribution-specific software for initial system

installation and configuration. For the users, most popular distributions offer mature application management systems that allow users to search, find, and install new applications with just a few clicks of the mouse. There are, at last count, over 350 distinct distributions of Linux.

Q. Outline four (4) transactional control commands in database.

Transactional Control Commands

In addition to the previously introduced categories of commands, there are commands that allow the user to manage database transactions.

1. COMMIT Saves database transactions
2. ROLLBACK Undoes database transactions
3. SAVEPOINT Creates points within groups of transactions in which to ROLLBACK
4. SET TRANSACTION Places a name on a transaction

Q. Discuss in detail what you understand by a pointer in C++

Simply stated, a pointer is an address. Instead of being a variable, it is a pointer to a variable stored somewhere in the address space of the program. It is always best to use an example to display a pointer. A pointer is simply the address of a memory location and provides an indirect way of accessing data in memory. A pointer variable is defined to 'point to' data of a specific type. For example:

```
int *ptr1; // pointer to an int

char *ptr2; // pointer to a char
```

The value of a pointer variable is the address to which it points. For example, given the definitions

```
int num;
```

we can write:

```
ptr1 = &num;
```

The symbol & is the address operator; it takes a variable as argument and returns the memory address of that variable. The effect of the above assignment is that the address of num is assigned to ptr1. Therefore, we say that ptr1 points to num.

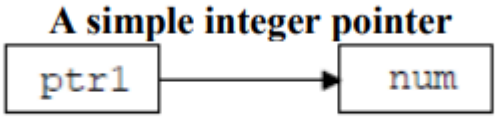


Figure 1.3(j): Assignment of the address of num to pointer ptr.

The following two rules are very important when using pointers and must be thoroughly understood.

1. A variable name with an ampersand in front of it defines the address of the variable and therefore points to the variable. You can therefore read line six as "pt1 is assigned the value of the address of "index".

2. A pointer with a "star" in front of it refers to the value of the variable pointed to by the pointer. Line nine of the program can be read as "The stored (starred) value to which the pointer "pt1" points is assigned the value 13". Now you can see why it is convenient to think of the asterisk as a star, it sort of sounds like the word store.

Q. Highlight in a tabular form (as specified in Table 1), the Windows XP commands that can be found in the applications menu.

Table 1.

S/N	Command	Function	Where it can be found/Location

S/N	**Command**	**Funtcion**	**Where it can be found or location**
1	Copy command	Copies the contents of an external storage device.	Found in the menu displayed by right-clicking the content to be copied.
2	Copy this file/folder command	Copies a selected file/folder to a new location.	Found in the File and Folder Tasks section of the My Computer window.
3	Delete this file/folder command	Moves a selected file/folder to the Recycle Bin	Found in the File and Folder Tasks section of the My Computer window.
4	Format command	Prepares an external storage devices to receive data.	Found in the menu displayed by right-clicking the icon of the device.
5	Make a new folder command	Creates a new folder.	Found in the File and Folder Tasks section of the My Computer window.
6	Move this file/folder command	Moves a file/folder to a new location.	Found in the File and Folder Tasks section of the My Computer window
7	Rename this	Selects the existing	Found in the File and

	file/folder command	file/folder name so that a new name can be typed.	Folder Tasks section of the My Computer window.
8	Restore command	Restores a file in the Recycle Bin to its previous location.	
9	Search for files/folders command	Used to find files and folders.	Found in the System Tasks section of the My Computer window.

Q. Write a short note on the SQL SELECT statement.

SQL SELECT STATEMENT

Apart from the use of SQL SELECT statement described in section 1.0, SELECT statement can also be used to restrict the columns that are displayed. The SELECT statement has the following capabilities:

i. Projection: Choose the columns in a table that are returned by a query. Choose as few or as many of the columns as needed.

ii. Selection: Choose the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.

iii. Joining: Bring together data that is stored in different tables by specifying the link between them.

The general syntax of the SQL SELECT STATEMENT is:

SELECT *|{[DISTINCT] column|expression [alias],...} FROM table

In its simplest form, a SELECT statement must include the following:

• A SELECT clause, which specifies the columns to be displayed

• A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause.

In the syntax:

SELECT is a list of one or more columns

* selects all columns

DISTINCT suppresses duplicates

column/expression selects the named column or the expression

alias	gives selected columns different headings
FROM table	specifies the table containing the columns

Q. Ada supports three (3) different types of strings that are designed to solve different problems. Analyse them illustrating with examples where necessary.

STRINGS

Ada supports three different types of strings and they are designed to solve a different problem. Every string type is implemented for each available Characters type (Character, Wide_Character, Wide_Wide_Character) giving a complement of nine combinations.

i. Fixed-length string handling

Fixed-Length Strings are arrays of Character, and of a fixed length. Since a fixed length string is an indefinite data type the length does not need to be known at compile time - the length may well be calculated at run time. In the following example the length is calculated from commandline argument 1:

```
X : String := Ada.Command_Line.Argument (1);
```

However once the length has been calculated and the strings have been created the length stays constant. Fixed-Length String Handling which allows padding of shorter strings and truncation of longer strings.

Exercise: Try the following example to see how it works:

File: show_commandline_2.adb

```
with Ada.Text_IO;
with Ada.Command_Line;
with Ada.Strings.Fixed;
procedure Show_Commandline_2 is
package T_IO renames Ada.Text_IO;
package CL renames Ada.Command_Line;
package S renames Ada.Strings;
package SF renames Ada.Strings.Fixed;
X : String := CL.Argument (1);
begin
T_IO.Put ("Argument 1 = ");
T_IO.Put_Line (X);
SF.Move (
Source => CL.Argument (2),
Target => X,
Drop => S.Right,
Justify => S.Left,
Pad => S.Space);
T_IO.Put ("Argument 2 = ");
T_IO.Put_Line (X);
end Show_Commandline_2;
```

ii. Bounded-length string handling

Bounded-Length Strings can be used when the maximum length of a string is known and/or restricted. This is often the case in database applications where only a limited amount of characters can be stored.

Like Fixed-Length Strings the maximum length does not need to be known at compile time - it can also be calculated at runtime - as the example below shows:

File: show_commandline_3.adb

```
with Ada.Text_IO;
with Ada.Command_Line;
with Ada.Strings.Bounded;
procedure Show_Commandline_3 is
package T_IO renames Ada.Text_IO;
package CL renames Ada.Command_Line;
function Max_Length (Value_1 : Integer;Value_2 : Integer)
return
Integer
is
Retval : Integer;
begin
if Value_1 > Value_2 then
Retval := Value_1;
else
Retval := Value_2;
end if;
return Retval;
end Max_Length;
pragma Inline (Max_Length);
package SB
is new Ada.Strings.Bounded.Generic_Bounded_Length (Max =>
Max_Length (Value_1 => CL.Argument (1)'Length,Value_2 =>
CL.Argument (2)'Length));
X : SB.Bounded_String
:= SB.To_Bounded_String (CL.Argument (1));
begin
T_IO.Put ("Argument 1 = ");
T_IO.Put_Line (SB.To_String (X));
X := SB.To_Bounded_String (CL.Argument (2));
T_IO.Put ("Argument 2 = ");
T_IO.Put_Line (SB.To_String (X));
end Show_Commandline_3;
```

iii. Unbounded-length string handling

Last but not least there is the Unbounded-Length String. If you are not doing embedded or database programming this will be the string type you are going to use most often as it gives you the maximum amount of flexibility.

As the name suggest the Unbounded-Length String can hold strings of almost any length – limited only to the value of Integer Last or your available heap memory.

File: show_commandline_4.adb

```
with Ada.Text_IO;
with Ada.Command_Line;
with Ada.Strings.Unbounded;
procedure Show_Commandline_4 is
package T_IO renames Ada.Text_IO;
package CL renames Ada.Command_Line;
package SU renames Ada.Strings.Unbounded;
X : SU.Unbounded_String
:= SU.To_Unbounded_String (CL.Argument (1));
begin
T_IO.Put ("Argument 1 = ");
T_IO.Put_Line (SU.To_String (X));
X := SU.To_Unbounded_String (CL.Argument (2));
T_IO.Put ("Argument 2 = ");
T_IO.Put_Line (SU.To_String (X));
end Show_Commandline_4;
```

Q. Describe the three (3) constructors that can be used to create an instance of the FileInputStream class.

The FileInputStream CLASS

The FileInputStream class performs file input operations, such as reading data from a file. The object of the FileInputStream class is used to read bytes of data from files. You can use the following constructors to create an instance of the FileInputStream class:

1. FileInputStream(File f): Creates a file input stream that connects to an existing file to be used as data source. The file object f specifies the name of the file to which an input stream is connected.

2. FileInputStream(String s): Creates a file input stream that connects to an existing file to be used as data source. The path of the file in file system is given by the string argument.

3. FileInputStream(FileDescriptor fdobj): Creates a file input stream that connects to an existing file to be used as data source. The fdObj file descriptor argument represents an existing connection to the specified file.

The BufferedInputStream CLASS

The BufferedInputStream class accepts data from other input streams and stores it in a memory buffer. The BufferedInputStream class reads data specified in any data format. A buffer stores the bytes of data and enables you to perform I/O operations on various data bytes simultaneously. The BufferedInputStream class supports read(), skip(), mark(), and reset() methods.

You can use the following constructors to create an instance of the BufferedInputStream class:

1. BufferedInputStream(InputStream is): Creates an input stream and adds a buffer to it. The default size of the buffer is 512 bytes.

2. BufferedInputStream(InputStream is, int size): Creates an input stream and adds a buffer of size specified by the size argument.

Q. What are random access files in Java?

Random access files enable you to access the contents of a file in a non-sequential manner. These files enable you to read and write text and bytes to any location in a file. These files implement the DataInput and DataOutput interfaces, which define the basic I/O operations.

Q. Enumerate the constructors used in creating an instance of the RandomAccessFile class in Java.

Constructors to create an instance of the RandomAccessFile class include:

1. RandomAccessFile(File fileObj, String mode): Creates an instance of the random access file.
2. RandomAccessFile(String name, String mode): Creates an instance of the random access file.

Q. Outline four (4) methods of the RandomAccessFile class

Methods of the RandomAccessFile class

1. void close() throws IOException: Closes the random access file and releases the system resources, such as streams and file pointers associated with the file.
2. long getFilePointer() throws IOException: Retrieves the current position of the file pointer in the specified file.
3. int skipBytes(int n) throws IOException: Ignores the number of bytes from a file specified by the n argument.
4. long length() throws IOException: Retrieves the length of the specified file
5. void seek(long position) throws IOException: Sets the current location of the file pointer at the specified position.

USING STREAMS IN JAVA

Input streams are the byte streams that read data in the form of bytes. The InputStream class is an abstract class that enables its subclasses to read data from different sources, such as a file and keyboard and displays the data on the monitor.

Methods of the InputStream class

1. int read(): Reads the next byte of data from an input stream. It returns –1 if it encounters the end of a stream.
2. int read(byte b[]): Reads the number of bytes from the array specified by the b[] argument. The read() method returns –1 when the end of the file is encountered.

3. int read(byte b[], int offset, int length) throws IOException: Reads the number of bytes from the array specified by the b[] argument. The argument offset specifies the starting position for the read operation, and length specifies the number of bytes to be read.
4. available(): Returns the total number of bytes available for reading in a stream.
5. long skip(long n) throws IOException: Ignores the specified number of bytes from an input stream.
6. mark(int nbyte): Places a mark at the current point in the input stream and this mark remains until the specified data bytes are read.
7. public final int readInt(): Retrieves an integer from the input stream.

Data: Variables and Constants

Constants

There are four basic types of constants in C. They are integer constants, floating-point constants, character constants and string constants

The following rules apply to all numeric-type constants.

1. Commas and blank spaces cannot be included within the constant.

2. The constant can be preceded by a minus (-) sign if desired. (Actually the minus sign is an operator that changes the sign of a positive constant, though it can be thought of as a part of the constant itself.)

3. The value of a constant cannot exceed specified minimum and maximum bounds. For each type of constant, these bounds will vary from one C compiler to another.

i. Integer Constants

An integer constant is an integer-valued number. Thus it consists of a sequence of digits. Integer constants can be written in three different number systems: decimal (base l0), octal (base 8) and hexadecimal (base 16).

ii. Floating-Point Constants

A floating-point constant is a base- 10 number that contains either a decimal point or an exponent (or both). Several valid floating-point constants are shown below.

0. 1 0.2 827.602

50000. 0.000743 12.3 31 5.0066

2 E-8 0.006e-3 1.6667E+8 .12121212e12

The following are not valid floating-point constants for the reasons stated.

1 Either a decimal point or an exponent must be present.

1,000.0 Illegal character (,).

2E+10.2 The exponent must be an integer quantity (it cannot contain a decimal point).

3E 10 Illegal character (blank space) in the exponent.

iii. Character Constants

A character constant is a single character, enclosed in apostrophes (i.e., single quotation marks). Several character constants are as follows; 'A' ' X ' ‘3’

iv. String Constants

A string constant consists of any number of consecutive characters (including none), enclosed in (double) quotation marks. Several string constants are as follows;

"green" "Washington, D.C. 20005H "

"$19.95" "THE CORRECT ANSWER IS:' '2* (I+3)/J "

"Line l\nLine 2\nLine 3"

Q. What is a variable?

Variables A variable is an identifier that is used to represent some specified type of information within a designated portion of the program. In its simplest form, a variable is an identifier that is used to represent a single data item; i.e., a numerical quantity or a character constant. A given variable can be assigned different data items at various places within the program. Thus, the information represented by the variable can change during the execution of the program. However, the data type associated with the variable cannot change.

Q. List three typical data type in C and their memory requirements.

DATA TYPES

C supports several different types of data, each of which may be represented differently within the computer’s memory. The basic data types are listed below. Typical memory requirements are also given. (The memory requirements for each data type will determine the permissible range of values for that data type. Note that the memory requirements for each data type may vary from one C compiler to another.)

Table 1.1 Data Types

Int	integer quantity	2 bytes or one word
Char	single character	1 byte
Float	floating-point number (i.e., a number containing a decimal point and or an exponent)	1 word **(4** bytes)
Double	double-precision floating-point number (i.e., more significant figures, and an exponent which may be larger in magnitude)	2 words (8 bytes)

Q. What do you understand by ‘’Java’’?

Java is an object-oriented program used to create Web pages with dynamic and interactive content, develop large-scale enterprise applications, enhance the functionality of World Wide Web servers (the computers that provide the content we see in our Web browsers), provide applications for consumer devices (such as cell phones, pagers and personal digital assistants) and for many other purposes. Java programs consist of pieces called classes. Classes consist of pieces called methods that perform tasks and return information when they complete their tasks. The class libraries are also known as the Java APIs (Application Programming Interfaces). The Java compiler, javac, is not a traditional compiler in that it does not convert a Java program from source code into native machine code for a particular computer platform. Instead, the Java compiler translates source code into byte codes. Byte codes are the language of the Java Virtual Machine—a program that simulates the operation of a computer and executes its own machine language.

Q. List and discuss the various kinds of variables in Java programming

The Java programming language defines the following kinds of variables:

1. Instance Variables (Non-Static Variables): Objects actually store their individual state in "non-static variables", that is, variables declared without the static keyword. Non-static variables are also known as instance variables because their values are unique to each instance of a class (to each object, in other words).

2. Class Variables (Static Variables): A class variable is any variable declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence; regardless of how many times the class has been instantiated.

3. Local Variables: Similar to how an object stores its state in variables; a method will often store its temporary state in local variables. The syntax for declaring a local variable is similar to declaring a variable (for example, int count = 0;). There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared - which is between the opening and closing braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.

4. Parameters: Recall that the signature for the main method is public static void main(String[] args). Here, the args variable is the parameter to this method. You may also occasionally see the term "member" used as well. A type's variables, methods, and nested types are collectively called its members.

Q. Data types in Java are the building blocks, give a detailed explanations of the various types of data types with appropriate examples

DATA TYPES

The data types in Java are the building blocks for the java program. Data types are portable across all computer platforms that support Java. This and many other portability features of Java enable programmers to write programs once, without knowing which computer platform will execute the program. Each data type has its size in bits. Data types includes; Boolean, Character, Floating point, and so on.

1. BOOLEAN TYPE

A Boolean can have only two possible values: True and False. A Boolean is used to express the results of logical operations. Boolean is also the type required by the conditional expressions that govern the control statements such as **if** and **for**.

Example 1.0// Demonstrate boolean values.

Output Generated:

b is false

b is true

This is executed.

10 > 9 is true

Q. Explain briefly the 4 integer types in Java programming

2. INTEGERS DATA TYPE

Java defines four integer types: byte, short, int, and long. All of these are signed, positive and negative values. Java does not support unsigned, positive-only integers.

- byte

The smallest integer type is byte. This is a signed 8-bit type that has a range from –128 to 127. Variables of type byte are especially useful when you're working with a stream of data from a network or file.

byte b, c;

- short

short is a signed 16-bit type. It has a range from –32,768 to 32,767. It is probably the least-used Java type

short s;

short t;

- int

The most commonly used integer type is int. It is a signed 32-bit type.

int lightspeed;

- long

long is a signed 64-bit type and is useful for those occasions where an int type is not large enough to hold the desired value. The range of a long is quite large.

long days;

4. FLOATING-POINT DATA TYPES

Floating-point numbers, also known as real numbers, are used when evaluating expressions that require fractional precision. There are two kinds of floating-point types, float and double, which represent single- precision (that uses 32 bits of storage.) and double-precision(uses 64 bits to store a value.) numbers, respectively.

float hightemp, lowtemp;

double pi;

5. CHARACTERS

In Java, the data type used to store characters is char. Unlike C++, in Java char is a 16-bit type. Java uses Unicode to represent characters; which defines a fully international character set.

char ch1, ch2;

ch1 = 88;

ch2 =’Y’;

Though chars are not integers, in many cases you can operate on them as if they were integers.

Q. Lists the various in-built packages of Java that you know

JAVA APPLICATION PROGRAMMING INTERFACE (API) PACKAGES

The Java API contains classes that are grouped in different packages in accordance to their functions, such as java.lang package provides various fundamental classes of Java. The java.util

package provides various utility classes, such as Date, Calendar, and Dictionary. The java.applet package provides the basic functions for creating applets.

The following table lists the various in-built packages of Java:

1. java.lang: Provides various classes, such as Object, System, and Class.

2. java.util: Provides various classes that support collection or groups of objects, such as hash tables, String parsing, and system properties.

3. java.io: Defines two streams, InputStream and OutputStream that determine the flow of bytes from a source to destination.

4. java.awt: Provides classes to implement graphical user interface, such as creating buttons, check boxes, text boxes, menus, and list boxes.

5. java.net: Provides classes that support network programming, such as Socket, ServerSocket, and DatagramSocket.

6. java.applet: Provides the Applet class that provides methods to display images, play audio files, and obtain information about the applet environment. Some of these methods are play(), getImage(), getAppletInfo(), and getAudioClip().

Q. List any 10 data types in COBOL

DATA TYPES IN COBOL

Standard COBOL provides the following data types

1. Character
2. Edited character
3. Numeric fixed-point
4. Binary
5. Numeric fixed-point packed decimal
6. Numeric fixed-point zoned decimal
7. Numeric floating-point
8. Edited numeric
9. Group (record)
10. Table (array)
11. Variable-length table
12. Renames (variant or union data)
13. Condition name
14. Array index

COBOL PROGRAMMING

Programming in COBOL in particular, is done by writing some simple COBOL programs that use the three main programming constructs - Sequence, Iteration and Selection. Any program consists of three main things;

1. The computer statements needed to do the job

2. Declarations for the data items that the computer statements need.

3. A plan, or algorithm, that arranges the computer statements in the program so that the computer executes them in the correct order.

Q. State any five name rules in COBOL

NAME RULES IN COBOL

All user-defined names, such as data names, paragraph names, section names condition names and mnemonic names, must adhere to the following rules:

1. They must contain at least one character, but not more than 30 characters.

2. They must contain at least one alphabetic character.

3. They must not begin or end with a hyphen.

4. They must be constructed from the characters A to Z, the numbers 0 to 9, and the hyphen.

5. They must not contain spaces.

6. Names are not case-sensitive: TotalPay is the same as totalpay, Totalpay or TOTALPAY.

HIERARCHICAL STRUCTURE OF COBOL PROGRAM

COBOL programs are hierarchical in structure. Each element of the hierarchy consists of one or more subordinate elements. The hierarchy consists of Divisions, Sections, Paragraphs, Sentences and Statements. We can represent the COBOL hierarchy using the COBOL metalanguage. A Division may contain one or more Sections, a Section one or more Paragraphs, a Paragraph one or more Sentences and a Sentence one or more Statements.

I. Divisions

A division is a block of code, usually containing one or more sections, that starts where the division name is encountered and ends with the beginning of the next division or with the end of the program text.

II. Sections

A section is a block of code usually containing one or more paragraphs. A section begins with the section name and ends where the next section name is encountered or where the program text ends.

Q. Using appropriate examples of each, describe the three categories of the loop structure in Ada

LOOP CONTROL STRUCTURES

Loops allow you to have a set of statements repeated over and over again as many times as you choice. There are three categories of the loop structure in Ada;

i. Endless loop

ii. Loop with a condition

iii. for loop

i. Endless loop

The endless loop is a loop which never ends and the statements inside are repeated forever. Never is meant as a relative term here — if the computer is switched off the loop will end.

*Endless*_Loop :

loop

Do_Something;

end loop *Endles*s_Loop;

ii. Loop with a condition

This loop can have a condition at the beginning (The statements are repeated as long as the condition is met. If the condition is not met at the very beginning then the statements inside the loop are never executed) or at the middle (the condition is within statements which has to be repeated it a criterion is met) or at the end (the statements are repeated until the condition is met and the check is at the end the statements which are at least executed once.). The syntax for the loop with on condition is as follows;

Loop with a Condition at the Beginning: *While_Loop* :

```
while X <= 5 loop
X := Calculate_Something;
end loop While_Loop;
```

Loop with a condition at the End: *Until_Loop* :

```
loop
X := Calculate_Something;
exit Until_Loop when X > 5;
end loop Until_Loop;
```

Loop with a condition at the Middle: *Exit_Loop* :

```
loop
X := Calculate_Something;
exit Exit_Loop when X > 5;
Do_Something (X);
end loop Exit_Loop;
```

iii. for loop

Often, you may need a loop where a specific variable is counted from a given start value up or down to a specific end value.

```
For_Loop :
for I in Integer range 1 .. 10 loop
Do_Something (I)
end loop For_Loop;
```

the need for a loop which iterates over every element of an array brings **for loop** to play most times. The following sample code shows you how to achieve this:

```
Array_Loop :
for I in X'Range loop
X (I) := Get_Next_Element;
end loop Array_Loop;
```

With X being an array, this is mostly used on arrays but it will also work with other types when a full iteration is needed.

Q. What are packages? Explain the parts of packages in Ada

PACKAGES IN ADA

Ada programming languages has well defined system of modularization and separate compilation. Even though Ada allows separate compilation, it maintains the strong type checking among the various compilations by enforcing rules of compilation order and compatibility checking. Packages are program units that allow the specification of groups of logically related entities. Typically, a package contains the declaration of a type along with the declaration of primitive subprograms (procedures or functions) of the type, which can be called from outside the package, while their inner workings remain hidden from outside users.

PARTS OF PACKAGES

There are 3 parts of a package but only the specification is necessary; the public package specification, private package specification and the package body. You can leave out the body and the private part if you don't need them.

I. The public package specification

This package specification of an Ada package describes all the subprogram specifications, variables, types, and constants etc that are visible to anyone who wishes to use the package.

```
package Public_Only_Package is

type Range_10 is range 1 .. 10;

end Public_Only_Package;
```

II. The private package specification

The private part of a package has two main jobs:

• To complete the defered definition of private types and constants.

• To export entities only visible to the children of the package

```
package Package_With_Private is

type Private_Type is private;

private type Private_Type is array (1 .. 10) of Integer;

end Package_With_Private;
```

III. The package body

The package body defines the implementation of the package. All the subprograms defined in the specification have to be implemented in the body. New subprograms, types and objects can be defined in the body that is not visible to the users of the package.

Q. In window XP environment, give examples of 5 settings that can be controlled using the control panel

1. Audio
2. Network
3. Computers graphics display
4. Themes
5. Printer setting
6. Date and time
7. Installations of programs

Q. Explain the rationale for the installation of BIOS prior to Linux operating systems installation?

BIOS provides the basic functions needed to boot your machine to allow your operating system to access your hardware. Before installing, you must ensure that your BIOS is setup correctly; not doing so can lead to intermittent crashes or an inability to install Linux. Many BIOS set-up menus allow you to select the devices that will be used to bootstrap the system.

Here's a road map for the steps you will take during the installation process.

1. Back up any existing data or documents on the hard disk where you plan to install.

2. Gather information about your computer and any needed documentation, before starting the installation and ensure that your computer will allow you to carry out the type of installation you want. Table 2.3(a) shows the types of requirements.

Table 2.3(a): Recommended Minimum System Requirements

Install Type	RAM	Hard Drive
No desktop	32 megabytes	400 megabytes
With Desktop	128 megabytes	2 gigabytes
Server	128 megabytes	4 gigabytes

3. Create partition-able space for Linux on your hard disk.

4. Locate and/or download the installer software and any specialized driver files your machine requires (except Linux CD users).

5. Set up boot tapes/floppies/USB sticks, or place boot files (most Linux CD users can boot from one of the CDs).

6. Boot the installation system.

7. Select installation language.

8. Activate the Ethernet network connection, if available.

9. Create and mount the partitions on which Linux will be installed.

10. Watch the automatic download/install/setup of the base system.

11. Install a boot loader which can start up Linux and/or your existing system.

12. Load the newly installed system for the first time.

Q. Why are control structures necessary in programming?

Computer programs are all about making decisions. If the user presses a key, the computer responds to the command. This section focuses on controlling program flow. Flow-control commands allow the program to decide what action to take based on the results of the C++ logical operations

performed. There are basically three types of flow-control statements: the branch, the loop, and the switch.

Q. List five (5) arithmetic operators in C language.

Arithmetic Operations

There are five arithmetic operators in C. They are

S/N	Operator	Purpose
1	+	addition
2	-	subtraction
3	*	multiplication
4	/	division
5	%	remainder after integer divisior

Q. List five (5) logical operators in C language.

Logical Operators

There are many logical operators in C. They are;

Operator	Meaning
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to
&&	and
||	or

Q. Define the following i) Variable ii) Function iii) Main function

i) Variable

A variable is an identifier that is used to represent some specified type of information within a designated portion of the program. In its simplest form, a variable is an identifier that is used to represent a single data item; i.e., a numerical quantity or a character constant. A given variable can be assigned different data items at various places within the program.

ii) Function

A function is a self-contained program segment that carries out some specific, well-defined task. C program can be modularized through the intelligent use of functions. the use of a function avoids

the need for redundant (repeated) programming of the same instructions. A function will process information that is passed to it from the calling portion of the program, and return a single value.

iii) Main function

The main() function is a section of code where the major instructions of a C program are placed. It contains the data that are declared as variables or constants, objects created from classes, namespace declaration, data type declaration and others. There are other functions that can be created in the program but the main() function must be created, without it there is no existence of the code.

Q. Briefly explain what you understand by a Class.

Class is at the core of Java. It is the logical construct upon which the entire Java language is built. It defines the shape and nature of an object. As such, the class forms the basis for objectoriented programming in Java. Any concept you wish to implement in a Java program must be encapsulated within a class. A class defines a new data type that can be used to create objects (an object is an instance of a class) of that type. A class is declared by use of the keyword class.

Java programs consist of pieces called classes. Classes consist of pieces called methods that perform tasks and return information when they complete their tasks. The class libraries are also known as the Java APIs (Application Programming Interfaces).

JAVA PACKAGES

A package is a set of classes that are stored in a directory, which has the same name as the package name. Packages enable you to organize the class files provided by Java. Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another. Because software written in the Java programming language can be composed of hundreds or thousands of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages. The Java packages are classified into the following two categories:

I. Java Application Programming Interface (API) packages: The Java API consists of various packages, such as java.lang, java.util, java.io, java.awt, java.net, and java.applet.

II. Java user-defined packages: The packages that a user creates are called user defined packages. The user-defined packages can be imported in any Java program.

Q. Mention the two (2) categories of Java packages

1. Java Application Programming Interface (API) packages
2. Java user-defined packages

Q. What is a method?

Methods (called functions or procedures in other programming languages) allow the programmer to modularize a program by separating its tasks into self-contained units. These methods are

sometimes referred to as programmer-declared methods. Methods are used for performing common mathematical calculations, string manipulations, character manipulations, input/output operations, database operations, networking operations, file processing, error checking and many other useful operations. The general form of a method is

```
type name(parameter-list) {

// body of method

}
```

Q. What is Windows 7 operating system?

WINDOWS 7 OPERATING SYSTEM

Windows 7 is an operating system. It controls the hardware of your computer and interprets the instructions from your software and operating systems to the hardware. When you use a menu command such as File , Save, Windows 7 is the driving force that writes the file to your disk or hard drive.

Windows 7 has a graphical user interface or GUI (pronounced “gooey”) that displays open applications and documents in areas on the screen called windows. Placing each open application into its own window allows Windows 7 to multitask. Multitasking is an operating system feature that allows more than one application to run at a time. When the Windows 7 operating system is running, the computer screen is referred to as the Desktop. Its desktop environment is attractive.

Q. Mention five (5) Windows 7 versions

There are several different versions of Windows 7 including:

1. Windows 7 Professional
2. Windows 7 HomeBasic
3. Windows 7 Premium
4. Windows 7 Starter
5. Windows 7 Ultimate

Q. Define and state the functionalities of Windows XP operating system.

WINDOWS XP ENVIRONMENT AND FEATURES

Windows XP has a graphical user interface or GUI (pronounced “gooey”) that displays open applications and documents in areas on the screen called windows. Placing each open application into its own window allows Windows XP to multitask. Multitasking is an operating system feature that allows more than one application to run at a time. When the Windows XP operating system is running, the computer screen is referred to as the Desktop: When the Windows XP operating system is running, the computer screen is referred to as the Desktop. Multitasking is an operating system feature that allows more than one application to run at a time. Windows XP handles multitasking by placing each running application into its own window.

Q. Define the following terms

i) Icon: Icons represent the applications and files available on the computer. Double-clicking an icon opens that file or application.

ii) Taskbar: The Taskbar displays buttons that represent each open file or application. In the example above, there are two open applications, My Pictures and WordPad. Clicking a button either displays or minimizes the window containing the file or application. The Taskbar also contains the start menu, Quick Launch toolbar, and the notification area.

iii) Quick lunch toolbar: The Quick Launch toolbar is used to start frequently used applications with just one click.

iv) Command button: A command button performs a command (makes something happen) when you click it. You'll most often see them in dialog boxes, which are small windows that contain options for completing a task

v) Slider: A slider lets you adjust a setting along a range of values. A slider along the bar shows the currently selected value. To use a slider, drag the slider toward the value that you want.

vi) Notification area: The notification area contains a clock and other icons that display the status of certain activities such as printing.

Q. How do you save a document and what are the procedures involved?

SAVING A DOCUMENT

Saving a document allows you to name it and to store it permanently on your computer's hard disk. That way, the document is preserved even when your computer is turned off, and you can open it again later.

To save a document:

1. Click the File menu, and click Save or Click the Save button

2. If this is the first time you are saving the document, you'll be asked to provide a name for it and a location on your computer to save it to.

Even if you've saved a document once, you need to keep saving it as you work. That's because any changes you've made since you last saved the document are stored in RAM, not on the hard disk. To avoid losing work unexpectedly due to a power failure or other problem, save your document every few minutes.

Q. List five (5) of the popular electronic devices that currently use Linux operating system.

An abbreviated list of some of the popular electronic devices Linux is used on today includes:

1. Dell Inspiron Mini 9 and 12

2. Google Android Dev Phone 1

3. HP Mini 1000 4

. One Laptop Per Child XO2

5. Volvo In-Car Navigation System

6. Motorola MotoRokr EM35 Phone

7. Yamaha Motif Keyboard

Q. Mention four (4) versions of Linux operating system.

THE VERSIONS OF LINUX

There are several versions of Linux and it includes;

1. Ubuntu
2. Debian
3. Mandriva
4. Red Hat
5. Slackware

Q. What is SQL?

SQL is the standard language for all databases. SQL is a nonprocedural language, in contrast to the procedural or third-generation languages (3GLs) such as COBOL and C that had been created up to that time. It is called a nonprocedural language because it describes what operation is to be performed rather how it is been performed. This unit studies the history of Structure Query Language and discusses the concepts of Relational Database Management System (RDMS) in relation to its use in SQL. It also highlights some basic commands and terminologies.

Q. Explain the history of the Structure Query Language

HISTORY OF SQL

The history of SQL began in an IBM laboratory in San Jose, California, where SQL was developed in the late 1970s. The initials stand for Structured Query Language, and the language itself is often referred to as "sequel." It was originally developed for IBM's DB2 product (a relational database management system, or RDBMS, that can still be bought today for various platforms and environments). In fact, SQL makes an RDBMS possible. Two standards organizations, the American National Standards Institute (ANSI) and the International Standards Organization (ISO), currently promote SQL standards to industry. The ANSI-92 standard is the standard for the SQL used throughout this book. Although these standard-making bodies prepare standards for database system designers to follow, all database products differ from the ANSI standard to some degree. In addition, most systems provide some proprietary extensions to SQL that extend the language into a true procedural language.

Q. Mention three (3) advantages of Structured Query Language.

THE ADVANTAGE OF SQL

SQL has the following advantages:

1. It is efficient
2. It is easy to learn and use
3. It is functional complete (with SQL, data in a table can be defined, retrieved and manipulated)

Q. Explain the history of databases

History of Relational Databases

Database systems store information in every conceivable business environment. From large tracking databases such as airline reservation systems to a child's baseball card collection, database systems store and distribute the data that we depend on. Until the last few years, large database systems could be run only on large mainframe computers. These machines have traditionally been expensive to design, purchase, and maintain. However, today's generation of powerful, inexpensive workstation computers enables programmers to design software that maintains and distributes data quickly and inexpensively.

The most popular data storage model is the relational database, which grew from the seminal paper "A Relational Model of Data for Large Shared Data Banks," written by Dr. E. F. Codd in 1970. SQL evolved to service the concepts of the relational database model. Dr. Codd defined 13 rules, oddly enough referred to as Codd's 12 Rules, for the relational model.

Q. Define some SQL terminologies

Database Terminologies

i. A Record: A record, also called a row of data, is each individual entry that exists in a table.

ii. A Column: A column is a vertical entity in a table that contains all information associated with a specific field in a table.

Iii. The Primary Key: A primary key is a column that makes each row of data in the table unique in a relational database. The primary key in the employees table is employee_id which is typically initialized during the table creation process. The nature of the primary key is to ensure that all product identifications are unique, so that each record in the employees table has its own employee_id. Primary keys prevents the possibility of a duplicate record in a table and they do not take null as values.

iv. A NULL Value: NULL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank. A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.

v. Field: Every table is broken up into smaller entities called fields. The fields in the employees table consist of employee_id, first_name, last_name, salary, hire_date, department_id. These fields categorize the specific information that is maintained in a given table. A field is a column in a table that is designed to maintain specific information about every record in the table.

Q. List and explain the basic SQL commands categories

SQL COMMANDS

i. Data Query Language

Though comprised of only one command, Data Query Language (DQL) is the most concentrated focus of SQL for modern relational database users. A query is an inquiry to the database for information. A query is usually issued to the database through an application interface or via a command line prompt.

The basic command is as follows:

- SELECT

ii. Data Manipulation Language

Data Manipulation Language, DML , is the part of SQL used to manipulate data within objects of a relational database.

There are three basic DML commands:

• INSERT

• UPDATE

• DELETE

iii. Data Definition Language

Data Definition Language, DDL, is the part of SQL that allows a database user to create and restructure database objects, such as the creation or the deletion of a table.

Some of the most fundamental commonly used DDL commands include the following:

• CREATE TABLE

• ALTER TABLE

• DROP TABLE

• CREATE INDEX

• ALTER INDEX

• DROP INDEX

• CREATE VIEW

iv. Data Control Language

Data control commands in SQL allow you to control access to data within the database. These DCL commands are normally used to create objects related to user access and also control the distribution of privileges among users.

Some data control commands are as follows:

• ALTER PASSWORD

• GRANT

• REVOKE

• CREATE SYNONYM

v. Data Administration Commands

Data administration commands allow the user to perform audits and perform analyses on operations within the database. They can also be used to help analyze system performance. Two general data administration commands are as follows:

• START AUDIT

• STOP AUDIT

vi. Transactional Control Commands

In addition to the previously introduced categories of commands, there are commands that allow the user to manage database transactions.

• COMMIT Saves database transactions

• ROLLBACK Undoes database transactions

• SAVEPOINT Creates points within groups of transactions in which to ROLLBACK

• SET TRANSACTION Places a name on a transaction

Q. What is a Pointer?

A pointer is simply the address of a memory location and provides an indirect way of accessing data in memory. A pointer variable is defined to ‘point to’ data of a specific type.

Q. What are control structures?

Statements in a program are executed one after the other in the order in which they are written. Control structures in Java are usually used to write complex programs require continuous execution till a particular output is achieved. Two groups of control structures are;

i. Selection Control Structure

ii. Repetition Control Structure

Q. Discuss the various types of control structures you know using appropriate examples in C++

1. SELECTION CONTROL STRUCTURE

A selection structure is used to choose among alternative courses of action in a program. There are two type of selection structure; the IF structure and the IF/ELSE structure.

I. IF SELECTION STRUCTURE

The if structure is a single-entry/single-exit structure. The if selection structure performs an indicated action only when the given condition evaluates to true; otherwise, the action is skipped.

```
if ( studentGrade >= 60 )

System.out.println( "Passed" );
```

II. IF/ELSE SELECTION STRUCTURE

The if/else selection structure allows the programmer to specify that a different action is to be performed when the condition is true rather than when the condition is false.

```
if ( studentGrade >= 60 )

System.out.println( "Passed" );

else

System.out.println( "Failed" );
```

2. REPETITION CONTROL STRUCTURE

A repetition structure allows the programmer to specify that an action is to be repeated while some condition remains true. There are three types of repetition structure; the while structure, the for structure and the do/while structure.

Q. Enumerate the factors that determines quality of printed pictures

Print quality depends on several factors including:

I. Choice of paper
II. Ink
III. Printer settings
IV. The quality of the original photo

Q. Enumerate the steps you would use to print pictures in windows 7

To print your own pictures

1. Double-click a picture to launch Windows Photo Viewer.

2. On the toolbar, click the Print button, and then click Print.

3. In the Print Pictures dialog box, select the printer, paper size, print quality, print style, and number of copies.

4. When you're done, click Print.

Q. What do you understand by a floating point constant?

A floating-point constant is a base- 10 number that contains either a decimal point or an exponent (or both).

Q. Give examples of two (2) valid and two (2) invalid floating point constants

Valid Floating Point Constants

I. 1 0.2 827.602
II. 0.000743 12.3 31 5.0066

Invalid Floating Point Constants

1 Either a decimal point or an exponent must be present.

1,000.0 Illegal character (,).

Q. Mention the rules guiding all numeric-type constants.

The following rules apply to all numeric-type constants.

1. Commas and blank spaces cannot be included within the constant.

2. The constant can be preceded by a minus (-) sign if desired. (Actually the minus sign is an operator that changes the sign of a positive constant, though it can be thought of as a part of the constant itself.)

3. The value of a constant cannot exceed specified minimum and maximum bounds. For each type of constant, these bounds will vary from one C compiler to another.

Q. What do you understand by Multidimensional Arrays?

Multidimensional arrays are actually arrays of arrays in Java, in this type of array; the rows are allowed to vary in length. An array may have more than one dimension (i.e., two, three, or higher). The organization of the array in memory is still the same (a contiguous sequence of elements), but the programmer's perceived organization of the elements is different. For example, suppose we wish to represent the average seasonal temperature for three major Australian capital cities.

Q. Briefly give 10 examples of Maths Methods in Java and state the function of each.

Table 1.4(a): Summary of the Maths Class Method in Java

Math class methods		
Method	**Description**	**Example**
abs(x)	absolute value of x	abs(23.7) is 23.7 abs(-23.7) is 23.7
ceil(x)	rounds x to the smallest integer not less than x	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0.0) is 1.0
exp(x)	exponential method e^x	exp(1.0) is 2.71828 exp(2.0) is 7.38906
floor(x)	rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is -10.0
log(x)	natural logarithm of x (base e)	log(Math.E) is 1.0 log(Math.E * Math.E) is 2.0
max(x, y)	larger value of x and y	max(2.3, 12.7) is 12.7 max(-2.3, -12.7) is -2.3
min(x, y)	smaller value of x and y	min(2.3, 12.7) is 2.3 min(-2.3, -12.7) is -12.7
pow(x, y)	x raised to the power y (i.e., x^y)	pow(2.0, 7.0) is 128.0
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0.0
sqrt(x)	square root of x	sqrt(900.0) is 30.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0.0

Q. What are Java Streams?

Input streams are the byte streams that read data in the form of bytes. The InputStream class is an abstract class that enables its subclasses to read data from different sources, such as a file and keyboard and displays the data on the monitor.

Q. Explain briefly what you understand by (i) Procedural (ii) structural (iii) object-oriented programming

(i) Procedural: A procedure, or function, is a set of specific instructions executed one after the other. The data was quite separate from the procedures, and the trick in programming was to keep track of which functions called which other functions, and what data was changed. To make sense of this potentially confusing situation, structured programming was created.

(ii) Structural: Structured programming remains an enormously successful approach for dealing with complex problems. By the late 1980s, however, some of the deficiencies of structured programming had become all too clear.

(iii) Object-oriented programming: Object-oriented programming attempts to respond to these needs, providing techniques for managing enormous complexity, achieving reuse of software components, and coupling data with the tasks that manipulate that data. The essence of object-oriented programming is to treat data and the procedures that act upon the data as a single object—a self-contained entity with an identity and certain characteristics of its own.

Q. What is BOOLEAN?

A Boolean in Ada is an Enumeration of False and True with special semantics. These types makes use of the logical and the relationnal operators on data; such as >, <= and so on.

FIXED POINT

A fixed point type defines a set of values that are evenly spaced with a given absolute precision. In contrast, floating point values are all spaced according to a relative precision. The absolute precision is given as the delta of the type. There are two kinds of fixed point types; ordinary and decimal.

1. Ordinary Fixed Point types: The delta gives a hint to the compiler how to choose the small value if it is not specified: It can be any power of two not greater than delta. You may specify the small via an attribute clause to be any value not greater than delta. (If the compiler cannot conform to this small value, it has to reject the declaration.) delta Delta range Low .. High

2. Decimal Fixed Point types: The small is defined to be the delta, which in turn must be a power of ten. (Thus you cannot specify the small by an attribute clause.) For example, if you define a decimal fixed point type with a delta of 0.1, you will be able to accurately store the values 0.1, 1.0, 2.2, 5.7, etc. You will not be able to accurately store the value 0.01. Instead, the value will be rounded down to 0.0.

If the compiler accepts your fixed point type definition, it guarantees that values represented by that type will have at least the degree of accuracy specified (or better). If the compiler cannot support the type definition (e.g. due to limited hardware) then a compile-time error will result. delta Delta digits Num_Digits

Q. What is RECORDS?

A record is a composite type that groups one or more fields. A field can be of any type, even a record. There are two types of records;

i. Basic Record: This type of record contains elements/data of different data types

ii. Null Record: This type of record does not contain any data type

Q. What is an array?

An array is a collection of elements which can be accessed by one or more index values.

Q. What is Conditional clauses (conditional control structures)?

Conditional clauses (conditional control structures) are blocks of code that will only execute if a particular expression (the condition) is true.