

ACHIEVER'S EDUCATION CONSULTANT
08141710097
SUMMARY OF CIT351 WITH PAST QUESTIONS

Course Marking Scheme.....

The course is divided into 5 modules and 15 study units. It will introduce students to fundamental concepts of C# Programming, . CIT 351 is all about, the textbooks and other materials to be referenced, what you are expected to know in each unit, and how to work through the course material. It suggests the general strategy to be adopted and also emphasizes the need for self assessment and tutor marked assignment.

There are also tutorial classes that are linked to this course and students are advised to attend. In the course of your studies, you will be equipped with definitions of common terms, characteristics and applications of object-oriented programming using C#. Finally, you will learn about C# expressions and operators. This course aims to introduce students to the basic concepts and features of C# programming.

Course Aim

Students should study them carefully before proceeding to subsequent units. Therefore, it may be useful to refer to these objectives in the course of your study of the unit to assess your progress. In this way, you can be sure that you have done what is required of you by the end of the unit.

[...]

Each unit contains some self assessment exercises and tutor marked assignments, and at some point in this course, you are required to submit the tutor marked assignments.

Systems: The Sixth Manifesto

The assignment file will be given to you in due course. In this file, you will find all the details of the work you must submit to your tutor for marking. The marks you obtain for these assignments will count towards the final mark for the course. Altogether, there are 15 tutor marked assignments for this course.

The presentation schedule included in this course guide provides you with important dates for completion of each tutor marked assignment. You should therefore endeavour to meet the deadlines. At the end of the course, you will need to sit for a final written examination. This examination will account for 60% of your total score.

Tutor Marked Assignments ix

You need to submit all the TMAs. The best 4 will therefore be counted. When you have completed each assignment, send them to your tutor as soon as possible and make certain that it gets to your tutor on or before the stipulated deadline. If for any reason you cannot complete your assignment on time, contact your tutor before the assignment is due to discuss the possibility of extension.

Extension will not be granted after the deadline, unless on extraordinary cases. Final Examination and Grading The final examination for CIT 351 will be of last for a period of 2 hours and have a value of 60% of the total course grade. The examination will consist of questions which reflect the self assessment exercise and tutor marked assignments that you have previously encountered. Furthermore, all areas of the course will be examined.

You might find it useful to review your TMAs and comment on them before the examination.

[...]

This table indicates the units, the number of weeks required to complete them and the assignments.

[...]

In distance learning, the study units replace the university lecturer. You are provided with exercises at appropriate points, just as a lecturer might give you an inclass exercise. The first item is an introduction to the subject matter of the unit and how a particular unit is integrated with the other units and the course as a whole. Next to this is a set of learning objectives.

These learning objectives are meant to guide your studies. The moment a unit is finished, you must go back and check whether you have achieved the objectives. If this is made a habit, then you will increase your chances of passing the course. The main body of the units also guides you through the required readings from other sources.

Self assessment exercises are provided throughout the unit, to aid personal studies and answers are provided at the end of the unit. Working through these self tests will help you to achieve the objectives of the unit and also prepare you for tutor marked assignments and examinations. You should attempt each self test as you encounter them in the units.

Read the course guide thoroughly

Note the time you are expected to spend on each unit and how the assignment relates to the units. You need to gather together all these information in one place such as a diary, a wall chart calendar or an organizer. Whatever method you choose, you should decide on and write in your own dates for working on each unit. Once you have created your own study schedule, do everything you can to stick to it.

If you get into difficulties with your schedule, please let your tutor know before it is too late for help. You will almost always need both the study unit you are working on and one of the materials recommended for further readings, on your desk at the same time. Work through the unit, the content of the unit itself has been arranged to provide a sequence for you to follow. As you work through the unit, you will be encouraged to read from your set books.

Keep in mind that you will learn a lot by doing all your assignments carefully. They have been designed to help you meet the objectives of the course and will help you pass the examination. Review the objectives of each study unit to confirm that you have achieved them. When you are confident that you have achieved a unit's objectives, you can start on the next unit.

Proceed unit by unit through the course and try to pace your study so that you can keep yourself on schedule. When you have submitted an assignment to your tutor for marking, do not wait for its return before starting on the next unit. When the assignment is returned, pay particular attention to your tutor's comments, both on the tutor marked assignment form and also written on the assignment. Check that you have achieved the unit objectives and the course objectives .

Tutors and Tutorials

There are 8 hours of tutorial provided in support of this course. You will be notified of the dates, time and location together with the name and phone number of your tutor as soon as you are allocated a tutorial group. Your tutor will mark and comment on your assignments, keep a close watch on your progress and on any difficulties you might encounter and provide assistance to you during the course. You must mail your tutor marked assignment to your tutor well before the due date.

At least two working days are required for this purpose. They will be marked by your tutor and returned to you as soon as possible. You should endeavour to attend the tutorials. This is the only opportunity to have face-to-face contact with your tutor and ask questions which are answered instantly.

You can raise any problem encountered in the course of your study. You will learn a lot from participating actively in discussions.

[...]

You are welcome to the world of C# programming language.

[...]

C# is a multi-paradigm programming language encompassing imperative, functional, generic, objectoriented , and component-oriented programming disciplines. It combines some of the best features of modern programming languages such as Java, C++ or Visual Basic. C# is one of the programming languages designed for the Common Language Infrastructure. It supports all features of an Object Oriented language such as

abstraction, encapsulation, inheritance, and polymorphism features.

Give a concise description of C#

Microsoft submitted C# to the ECMA standards group mid-2000. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg, the designer of Borland's Turbo Pascal. It has an objectoriented syntax based on C++.

Microsoft had considered keeping the name «Cool» as the final name of the language, but chose not to do so for trademark reasons. July 2000 Professional Developers Conference, the language had been renamed C#, and the class libraries and ASP.NET runtime had been ported to C#. C#'s principal designer and lead architect at Microsoft is Anders Hejlsberg, who was previously involved with the design of Turbo Pascal, CodeGear Delphi , and Visual J++. This is similar to the language name of C++, where «++» indicates that a variable should be incremented by 1.

The sharp symbol also resembles a ligature of four «+» symbols , further implying that the language is an increment of C++. This convention is reflected in the ECMA334 C# Language Specification. However, when it is practical to do so, Microsoft uses the intended musical symbol.

Some notable distinguishing features of C# are

Static members of public classes can substitute for global variables and functions. Local variables cannot shadow variables of the enclosing block, unlike C and C++. Variable shadowing is often considered confusing by C++ texts. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack.

Multiple inheritance is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication, avoid dependency hell and simplify architectural requirements throughout CLI. Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default. Enumeration members are placed in their own scope.

C# provides properties as syntactic sugar for a common pattern in which a pair of methods, accessor and mutator encapsulate operations on a single attribute of a class. In this unit, we defined some basic concepts of C#. We also looked at the advancements in C# as well as the language name. We hope you enjoyed this unit.

Outline at least 5 distinguishing features of C#.

What it does not specify

In unit 1, we gave an overview of 'C# programming' as well as its basic features.

This International Standard does not specify

All minimal requirements of a data-processing system that is capable of supporting a conforming implementation. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. The language is intended for use in developing software components suitable for deployment in distributed environments. Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

From our studies in this unit, it is vital to remember that the ECMA

We hope you found the unit enlightening.

Outline the aims of ECMA

The initial task we have in this unit is to identify the target platform of C#, so that you can see the relationship between C# and . You might be tempted to think you're never going to get to grips with this. But don't worry - after a few lessons, things will start to feel familiar, and you will gain more confidence. At the time of this writing the .

NET Framework runs on Windows operating systems, and I know of two other major operating systems for which a version of the . NET Framework is being developed. So you can see that although C# is designed to target only the Framework, the Framework itself is flexible enough to run your C# programs on many types of systems. First, C# is not the only language that can be used to write .

C# is not the only language that can be used to write. In this unit, we considered the .

Identify the categories of C# type system

CSharp is a strongly typed language. We equally saw that the key difference between value types and reference types is the mode of storing values in memory. In this unit, we learnt about the general notion of type and the C# type system in particular.

[...]

Unboxing is the operation of converting a value of a reference type into a value of a value type. Unboxing in C# requires an explicit type cast. With Boxing and unboxing one can link between value-types and reference-types by allowing any value of a value-type to be converted to and from type object. Boxing and unboxing enables a unified view of the type system wherein a value of any type can ultimately be treated as an object.

State 2 applications of boxing and unboxing?

All types including value types derive from the type object. It is possible to call object methods on any value, even values of primitive types such as int. An int value can be converted to object

and back again to int. When an object box is cast back to its original value type, the value is copied out of the box and into the appropriate storage location.

Boxing a value of a value-type consists of allocating an object instance and copying the value-type value into that instance.

G_Box {

This is different from a conversion of a reference-type to type object, in which the value continues to reference the same instance and simply is regarded as the less derived type object. An unboxing conversion permits an explicit conversion from type object to any value-type or from any interface-type to any value-type that implements the interface-type. unboxing conversion of an object box to a value-type G consists of executing the expression .value. InvalidCastException is thrown.

Unboxing is the operation of converting a value of a reference type into a value of a value type. In summary, this unit looked an essential concept of C# type system boxing and unboxing. We equally identified the relevance of boxing and unboxing conversions.

[...]

CSharp is a strongly typed language, therefore every variable and object must have a declared type. We often refer to C# as being a strongly typed language. When we declare a variable, we have to tell the compiler about what type of the data the variable can hold or which data type the variable belongs to.

CSharp default initialization value : 0

In conclusion, CSharp is a strongly typed language, therefore every variable and object must have a declared type. In this unit, we discovered what data types portray and identified how to declare a variable in C#. Hope you grasped the key points.

This unit describes the fundamental naming conventions in C#

Much of the naming standards are derived from Microsoft's .

What is the relevance of using the correct conventions when naming objects?

Namespaces are named using Pascal Case with no underscores. This means the first letter of every word in the name is capitalized. If an assembly contains only one namespace, they should use the same name. Otherwise, Assemblies should follow the normal Pascal Case format.

Pascal Case, no underscores or leading «C», «cls», or «I». Classes should not have the same name as the namespace in which they reside. Any acronyms of three or more letters should be pascal case, not all caps. Try to avoid abbreviations, and try to always use nouns.

Net 2.0, all classes should inherit from the System. Exception base class, and not inherit from the System. Try to avoid abbreviations. Many programmers have a nasty habit of overly abbreviating everything.

This should be discouraged. Here is an example of a class that uses all of these naming conventions combined. Winding up, we can go over the key points of this unit. The naming standards are derived from Microsoft's .

We equally saw an example of a class that uses all the naming conventions. We hope you have found this unit interesting. This unit describes the basics of how the applications you write will be interpreted.

After going through this unit you would be able to

A statement can declare a variable, define an expression, perform a simple action by calling a method, control the flow of execution of other statements, create an object, or assign a value to a variable, property, or field. Statements can be grouped into comma-separated statement lists or brace-enclosed statement blocks.

What is the significance of statements?

Among other purposes, code blocks serve to limit the scope of variables defined within them. Code blocks can be nested and often appear as the bodies of methods.

DoSomeWork; // limitedVariable is no longer accessible here

} // Here ends the code block for the body of «MyMethod». Single-line comments, as one would expect, end at the first end-of-line following the «//» comment marker. //This style of a comment is restricted to one line. This comment is used to generate XML documentation.

This is the most recommended type. Avoid using butterfly style comments. C# is case-sensitive, including its variable and method names. Writing the following code would result in a compiler error unless an object named console had been previously defined.

The following corrected code compiles as expected because it uses the correct case

This unit provided an overview of C# syntax. It equally outlined the styles of comments allowed in C#. We hope you found this unit enlightening. Hello world Source Code C# and .

[...]

In the previous unit we examined C# syntax. Here, we will be looking at Hello World a-la C# and the minimal requirement to use C#.

State the minimal requirements for using C#

They are often used when referring to .NET code generically, as opposed to referring to a specific programming language like C# or Visual Basic. Hello World Source Code The source code in Figure 1-1 displays the text «Hello World!» in a window. In this example, the MyForm class is defined to derive its functionality from the Form class .

All C# applications must have a static method named Main defined to be the entry point of the application. This means that it would see methods named Main and main as different methods. The entry point method for C# programs must be called Main. The static Main method can be defined in any class in the application, so long as its name is “Main” and it is declared to be static.

They are similar to global functions, and are often referred to as type-methods . The OnPaint method is an override of a virtual method on the Form class. It is called when the window needs to paint itself.

We mentioned in the introduction that a big requirement of C#

Programming was learning the . This is true. We will make references to reusable object types defined in the FCL throughout this tutorial.

To further emphasize the importance of the FCL, look back at Figure

While the code is written in C#, a great deal of the functionality comes from types defined in the FCL. For example, the window behaves as it does because MyForm is derived from the FCL class Form. Additionally, the EventArgs, Graphics, Font, and Brushes classes play an important role in the drawing of the string on the form. And even the «Hello World!» string itself is of type String.

All of these types are defined by the FCL, and useable within your C# applications. First, types defined in the FCL exist in a hierarchy of namespaces. For example the Form class is really the System. Notice that the code in Figure 1-1 has two using statements.

Second, the types in the FCL are published in files called Assemblies. When you compile your C# application you must include references to the assemblies that are used by your code. .NET lets you add references to your projects, and the command line C# compiler uses the /R switch to indicate referenced assemblies. We will consider more about building your code later.

Minimal Requirements to Use C#

In this section we are going to consider how to build a C# application. More specifically, we are going to see the minimal requirements to use C#. The most common way of programming with C# is by using the Visual Studio .NET developers integrated development environment.

First, you should install the latest version of the .NET Framework to your PC running Windows. The C# compiler is called csc.exe and exists in the following directory. This lets me type csc from the command line in any directory in my file-system.

This is all you need to use C#. You can use any editor you like to create C# source code modules. You should give your C# modules a .Cs extension, and you are good to go.

If you were to cut and paste the source code from Figure 1-1 into a file, and save it as HelloGui.cs, you could then compile it into an executable with the following command.

The source code in Figure

Compile the source code using the command line compiler or the Visual Studio IDE. ♣ Outline the steps involved in building and running Hello GUI.

Console Application

In the previous modules, we discussed mainly the theoretical aspects of C# programming. This unit describes the process of creating console assemblies. Outline the steps required to build and run a console application Console Applications Console applications are the kind of executables that you would use to create a command line utility. The command line utility could be dir or xcopy.

Actually many ad-hoc C# projects are console applications.

What are console applications?

Figure 1 Rabbits.cs The C# source code shown in Figure 1 will calculate how many rabbits you get after a certain number of generations . This is because the default assembly built by the C# compiler is a console executable. Rabbits.cs sample also shows some interesting things about C# in general, so we will highlight them here. Like our last application, Rabbits.cs defines a class which defines an entry point function named Main.

The Main method in Rabbits.cs does something a little different by taking a parameter defined as an array of String objects. These are the command line arguments passed to the program. Rabbits.cs uses structured exception handling to handle errors. This is the try-catch code block that surrounds the bulk of the application.

The while loop syntax in Rabbits.cs, is exactly what it would be in C, C++, or Java. The sample code in Figure 1 uses the static WriteLine method of the Console type defined in the Framework Class Library to output text to the console window. Notice that an instance of the Console type was not necessary to call the method. This is because WriteLine is defined as a static method.

The Decimal type is well suited for financial and scientific applications. Its' written this way just for fun. There is nothing about C# that would prohibit you from implementing this algorithm using recursion. In this unit, we considered the concept of console application, the types of command line utility and a classic example of a console application.

We equally identified the steps required to build and run a console application.

In this unit, you'll gain knowledge of how to create GUI assemblies in

The source code in Figure 1 seems complex. Like the previous example, Tribbles.cs shows a lot more than just how to compile source code. Button and ListBox, as well as exception handling. An App class which has a static Main method to be used as the programs entry point, and the TribbleForm class which is derived from form and implements the programs window.

Most of the programs functionality exists in the TribbleForm class'

Although you can write GUI code from scratch, it is far more common to use a forms designer to help with the task. Both allow you to design your GUI using graphical tools, and the designer will create the code. C# GUI programs use a set of classes in the FCL known as the Windows Forms classes. Finally we learnt about the steps required to build and run a GUI application.

In this unit, you have a chance to learn another aspect of C#

We will study the procedure for building a code library. We'll equally look at the syntax for compiling objects. Knowing how to create executables with C# is important. But component development is going to become increasingly more important as the software industry evolves.

More and more application code is going to be housed in reusable objects. In fact, when you design web pages using C#, the entire code that implements the web page exists in a library assembly, with one or more objects derived from the Page type defined in the FCL. But we should start with a more mundane object library. We admit that the Fib object is perhaps one of the strangest implementations of a Fibonacci generator in existence but it does demonstrate reusable object libraries in C# nicely.

List 3 types of static properties public class Fib{

Figure 1 FibObj.cs The source code in Figure defines an object named Fib that can be used from other assemblies. This will create an assembly named FibObj.dll. FibObj.dll does not have a static Main entry method defined, and it can not be executed directly. If you try to build a non-library assembly without an entry point method defined, the compiler will give you an error message.

We have now built a binary library containing an object with executable code. Figure 2 FibTest.cs The sources for FibTest.cs can be used to test the FibObj.dll. However, special measures must be taken when compiling the sources in Figure 2. First, the referenced file is the binary assembly file FibObj.dll, not the source code file FibObj.cs.

Second, the lack of /Target switch indicates to the compiler that

FibTest.cs should be compiled into the default assembly type, which is a console application. Every time you use a type defined by the FCL you are referencing an assembly that

ships with the Framework Class Library. What is happening here between FibTest.exe and FibObj.dll is the simple foundation of a very important ability. That is the ability for objects to use each other without being compiled together .

The source code in Figure 1 FibObj.cs is a C# reusable object to be compiled into a library. The source code in Figure 2 FibTest.cs is a brief console application designed to consume or test the object in FibObj.dll. Outline the procedure for building a code library Microsoft Visual C#.

Expressions and Operators CREATING A PROJECT

NET manages a lot of the details of project management, building, debugging, and GUI development. Most programmers actually spend about 50% of their C# programming time using Visual Studio. IDE's commonly include source code editors , as well as project managers, online-help and debuggers, all in one software package. The idea is to keep all of your programming related tasks in one place.

When you run VS.NET you are presented with a Start Page. This page has links that help you create a new project or open an existing one. VS.NET projects are arranged in a hierarchy of projects and solutions. Solutions are sort of a collection of projects.

Most often you will have one project per solution, but if you are creating a program with an executable and a library of reusable classes, then you might make a solution with two projects, for example.

What is the main role of an IDE?

When you select new project you are presented with a screen as follows. The type of project that you will select from the pane on the left is Visual C# Projects . Then you will build your project from a template on the right. Windows Application – This template sets up a project for building a GUI application.

Console Application – If you want to create a console or command line application you should use this template. Class Library – If you want to make a library of reusable types, you should use the Class Library template. The assembly that you build from this project will not be executable, but you can use the types in this file from other executables. ASP.NET Web Application – This template creates a web forms application for creating web-pages with C#.

I will discuss web forms in detail in a tutorial later in this series. This is an advanced project type. I will discuss web services in detail in a tutorial later in this series. This includes editing source code files, compiling, and debugging.

The general rule is that you right-click your mouse on the item you want to affect, and then select the action from the menu that pops up. Cs source file to the project FibObj shown in Figure2, you could simply right click on the icon for FibObj, and then select Add>Add New Item. If you choose to program in C# without using Visual Studio. NET, you can write all the same kinds of software that you can write with VS.Net.

If you do choose to use Visual Studio. NET, I would suggest setting aside a couple of hours to just create several trial projects. Build them, debug them, and try making minor modifications to them. This will introduce you to the look and feel of VS.NET and will make it more productive for you when you have to do real work with the tool.

IDE's commonly include source code editors, as well as project managers, online-help and debuggers, all in one software package. You may now proceed to the tutor marked assignment below.

In this unit, we'll learn about the syntax ground rules and the primitive

List the common C# primitive types and their corresponding aliases The following are some basic rules that apply to C# code syntax. This means that method and variable names are distinguished by their case. For example, a method named Hello and a method named hello would be seen as different methods in C#. ♣ White space means nothing in C#.

The following two code blocks are seen as exactly the same by the C# compiler.

if{DoSomething}

You should use white space to increase the readability of your source code. The only exception is the end of a code block , which does not require a terminating semicolon . ♣ Statements can be grouped into a single unit called a code block. This is done by putting the statements inside of a pair of curly braces .

Code blocks can be nested. Outline at least three basic rules that apply to C# code syntax The table shown later in this section shows the types in the Framework Class Library that are treated as primitive types by C#. Primitive types are special because the C# language will allow direct assignment, math operations, and comparison of primitive types.

One other interesting point about the primitive types in C# is that the C#

Choosing which name to use in your source code is a matter of style. The most important thing is that you recognize that the FCL name for a primitive type and the alias name for a primitive type both refer to the same underlying type.

[...]

C# has a String type, rather than requiring that strings be represented as an array of Char variables like C or C++. C# has a Boolean type, and it is not valid to use a numerical type where a Boolean expression is required. C# also includes two keywords for indicating true and false. Since C# uses types defined in the FCL as its primitive types, C#'s primitive types are Object's just like all other types in the system.

The primitive variables are shown in red. Figure 2 Primitive Code Snippet The code in Figure 2

is the first in this tutorial to not be a compile compileable C# module. C# uses types defined in the FCL as its primitive types, C#'s primitive types are Object's just like all other types in the system. In this unit, we identified the basic rules that apply to C# code syntax. We also discovered the concept of primitive types in C# and the common C# primitive types as well as their corresponding aliases. You may now proceed to the tutor marked assignment.

List at least 5 primitive types in C# and their corresponding aliases.

In this unit you learn about the common expressions and operators in

Expressions like the following are common in programming, and are also common with C#. C# is a very type safe language, however, and expressions are strongly typed. The following code would be valid in C, but is an error in C#, because a Boolean expression is expected, and an Int32 expression is provided.

[...]

In this unit, we were made to understand that expressions are an integral part of programming, and C# treats expressions much the same as many languages. We equally discovered that operators such as +, *, and / are used to manipulate or affect the data in variables, and operators have precedence. Operator precedence can be modified using parenthesis.



**NATIONAL OPEN UNIVERSITY OF NIGERIA,
PLOT 91 CADASTRAL ZONE, NNAMDI AZIKIWE EXPRESSWAY, JABI – ABUJA
FACULTY OF SCIENCES**

NOVEMBER 2018 EXAMINATION

COURSE CODE: CIT 351

COURSE CREDIT: 2

COURSE TITLE: C# PROGRAMMING

TIME ALLOWED: 2 Hours

INSTRUCTION: ANSWER QUESTION (1) AND ANY OTHER THREE (3) QUESTIONS

QUESTIONS

1. Study the figure provided and answer the questions that follow:

```

using System;

public class Fib{
    Decimal current;
    Decimal last;
    public Fib(){
        current = 1;
        last = 0;
    }

    private Fib(Decimal last, Decimal secondToLast){
        current = last+secondToLast;
        this.last = last;
    }

    public Fib GetNext(){
        return new Fib(current, last);
    }

    public Decimal Value{
        get{return current;}
    }
}

```

- 1a. Identify and name the object within the source code. (2 marks)
- 1b. Write the command line for compiling this object. (4 marks)
- 1c. Name any 2 operators in this source code. (4 marks)
- 1d. Assuming you are required to build a C# application on a notebook running Windows, state the minimal requirements to be met prior to building this application on the notebook.. (11 marks).
- 1e. How are naming standards derived in C# Programming? **(4 marks)**
- [Total = 25 marks]**
- 2a. Generally, VisualStudio.NET is regarded as an Integrated Development Environment. State five (5) good reasons to support this statement. (10 marks)
- 2b. Write down the command for building a console application named **PEACE.cs** within the hard drive. (2 marks).
- 2c. Give a concise description of the concept of case-sensitivity in C# programming

language.

(3 marks)

[Total = 15 marks]

3a. Give a concise description of how memory leaks are handled in C# programming.

(1 mark)

3b. Write down the procedure of grouping 'statements' in C# Programming. (8 marks)

3c. State the three (3) main categories of Static properties commonly employed in C# programming.

(6 marks)

[Total = 15 marks]

4a. Outline the procedure required to generate and run an application saved as

Technology.cs

(8 marks)

4b. State the standard rule for accomplishing the following tasks in C# Programming:

Designating an end statement.

(2 marks)

Enhancing source code legibility

(2 marks)

4c. List the three (3) permissible forms of comments allowed in C#.

(3 marks)

[Total = 15 marks]

5a. Explain the statement 'C# is a programming language designed for the Common Language Infrastructure.

(4 marks)

5b. Give a short explanation of the following C# Primitives:

String

(1 mark)

Int64

(1 mark)

SByte

(1 mark)

Char

(1 mark)

Object

(1 mark)

5c. List three (3) core tasks that can be performed in a C# project within the VisualStudio.NET setting.

(6 marks)

[Total = 15 marks]