

I. (20 pts.) True or False Questions

- ~~False~~ In C++, when array variable is used as a parameter, the communication is never by ~~value~~.
- ~~False~~ If a given function in a modular program has a syntax error, the error will not be reported by the compiler as long as the function is not called.
- ~~True~~ Thanks to scope rule 2, programmers can use same identifier for different purposes in the same function.
- ~~True~~ An actual parameter cannot be a **constant** if the corresponding formal parameter is an alias.
- ~~False~~ If a function is re-defined in a nested scope within a given modular program, it's the inner definition that takes precedence over the outer.
- ~~False~~ A function can be declared within another function as long as a prototype is provided above the function.
- ~~True~~ The physical placement of functions in a modular program dictates their order of execution.
- ~~True~~ If a function has a potential run time error, then the error occurs only if the function is called directly or indirectly by the main function.
- ~~False~~ A called function must always be physically placed on top of the calling function.
- ~~False~~ A function call can be used as an actual parameter only when it is a **non-void (or typed)** function.

II. (20 pts) Multiple Choice Questions.

- Which of the following is **false** about the **return** statement
 - It may be used to stop a "monolithic" program
 - It may be used to pass both data and control to the calling function
 - ~~It need not always return a value to the calling function~~
 - It is never used to pass data from the calling to the called
 - None of the above are false about the **return** statement
- One of the following happens if the size of an array in a formal parameter list of a function is missing
 - Only one memory space (of appropriate size) is reserved for the array irrespective of the size
 - The actual size of the memory space to be reserved for the array parameter will be determined at calling
 - ~~A compile time error is generated~~
 - A run time error will surely occur if and when the function is called
 - ~~All of the above do not happen~~
- One of the following is **not** a necessary condition for a function to call another. Assume that none of the functions are the **main** function
 - The called function must be above the calling one
 - Data must be communicated at least in one direction
 - Both functions must have identical number and type of parameters (if any)
 - At least one of them must be called directly by the **main** function
 - ~~All are not necessary conditions~~
- Which of the following is true about inter module communication using the **function identifier** (12)
 - It allows to send data only from calling to called
 - The calling function must be **typed (non-void)**
 - ~~The called function must have a return statement~~

	D. The returning value cannot be a pointer E. All of the above
15.	All but one of the following is allowed (if done in same program) thanks to Scope Rules A. Same identifier declared as a function name and as a local variable in the same function ∞ ☒ B. Same identifier declared as a common variable and as a function name ☒ C. Same identifier declared as a common variable and as a parameter to a function D. Same identifier declared as function name and as a parameter of the function $\times$ E. All of the above are not allowed
16.	Which of the following is not true regarding data transfer mechanisms in C++? A. Constant pointer to a constant data is the best data transfer mechanism when we want the security feature of pass by value and the efficiency of reference arguments. B. Despite the problem of securing the data, common variables are as efficient as parameter passing by reference. $\checkmark$ ☒ C. Practically, it is possible to use parameter passing by value in place of parameter passing by reference in all cases. $\checkmark$ D. Parameter passing by reference can totally replace the use of parameter passing by value, thanks to 'const' modifier. E. None
17.	Which of the following cannot legitimately be passed to a function A. a constant B. a structure ☒ C. a header file D. a variable E. all can be passed
18.	One of the following is false about a variable declared at the Program Scope (global variable) A. It can be used as a parameter to any of the functions $\checkmark$ ☒ B. It can be re-declared within one of the functions if need be but to the same type ☒ C. It is initialized to zero at the beginning of the program execution D. It can be used to exchange data between the different functions of the program E. A & C
19.	Which one of the following is true about scope rule 1 ☒ A. It allows different teams to share same identifiers for different local purposes $\checkmark$ ☒ B. It allows similar identifiers to be declared in the same module as long as they are not of the same type C. It allows similar identifiers to be declared in the same module as long as they represent different objects (i.e. Variables, constants, types, functions, or enumerated elements) D. It allows global identifiers to be re-declared in any of the functions E. All are true
20.	One of the following is false about modular programs in C++ A. The sequence of execution of functions in a modular program do not depend on their physical placement B. It is possible to pass data from one function to another even if they don't call each other directly C. Two communicating functions cannot protect their common data as long as they use common variables (i.e. T1) ☒ D. A typed function can receive data through its identifier from a calling function E. All are true

III. (20 pts) Short Answer Questions 1.

For each of the modules below, describe the incoming and outgoing data and then provide a prototype. Do not write the code!

(2 pts) A module, named **swap** that swaps two real numbers.

Incoming: x, y

Outgoing: x, y

$x \leftrightarrow y$

Prototype: $\text{void swap}(\text{int}^* x, \text{int}^* y)$

(3 pts) A module, named **addAllPrimes** that adds and returns the sum of all the prime numbers below a given natural number.

Incoming: n

Outgoing: int

Prototype: $\text{int addAllPrimes}(\text{int} n)$

(5 pts) A module, named **suppressVowels** that suppresses (deletes) all the vowels found in a line of text.

Incoming: $\text{const char}^* \text{text}$

Outgoing: char^*

Prototype: $\text{char}^* \text{suppressVowels}(\text{const char}^* \text{text})$

(5 pts) A module, named **highestFrequentLetter** that identifies the highest frequent letter in a given line of text.

Incoming: $\text{const char}^* \text{text}$

Outgoing: char

Prototype: $\text{char} \text{highestFrequentLetter}(\text{const char}^* \text{text})$

25. (5 pts) A module, named **identically**, that tells whether or not two lists of same size are identical. You may consider the list to contain integers.
Incoming: ~~identically~~ **identically** **identically**

Outgoing:

identically **identically** **identically**

Prototype: **bool identically(int a[], int b[], int n)**

08 IV. (30 pts) Short Answer Questions 2.

All the questions in this part are based on the function **extractWords** that extracts and counts all the words that maybe found in a given line of text. For the sake of simplicity, the line of text is assumed to have no punctuation marks. Line numbers are added to facilitate the reading of the source code.

```

1. const SPC      = ' ', // the SPACE character
2. const MAXNUM   = 20; // maximum nb of words in a sentence
3. const MAXLEN   = 30; // maximum length of each word
4. void extractWords(const char * mystr, char extWd[MAXLEN], int & nbWords)
5. { char * mysent, * spcptr;
6.   int i;
7.   strcpy(mysent, mystr);
8.   strcat(mysent, " "); // to extract the last word
9.   i = 0;
10.  while (strlen(mysent) != 0)
11.  { while ( strlen(mysent) && (mysent[0] == SPC) ) mysent++;
12.    spcptr = strchr(mysent, SPC);
13.    if (spcptr != NULL) {
14.      strcpy(extWd[i], mysent, spcptr - mysent);
15.      extWd[i][spcptr - mysent] = '\0';
16.      i++;
17.      mysent = spcptr + 1;
18.    }
19.  }
20.  nbWords = i;
21. }
```

1. Identify the local variables	int i; ✓
2. Identify the pre-defined string functions used	char * mysent, * spcptr
3. Which formal parameter is used to bring the line of text from the calling function	char * mystr ✓

4.	Which formal parameter is used to transfer the extracted words to the calling function	πP_{023} α
5.	Which formal parameter is used to transfer the number of words information to the calling function	~~πP_{023}~~ πP_{21} α
6.	List the protocol(s) of communication used	~~πP_{023}~~ πP_{023} α
7.	List the tools of communication used	~~πP_{023}~~ πP_{023} α
8.	What is the purpose of line # 7	to copy words to mySent α
9.	What is the purpose of the keyword const in the first formal parameter	to keep the value α
10.	Which line of code actually searches for the next word to extract	15, 16, 17 α
11.	(4 pts) If the number of words information is to be sent thru the function identifier (ie. T2), rewrite the declaration (only) of extractWords to reflect this change	(~~char MAXLEN~~ , char myStr) α
12.	(6 pts) Write a function (could be the main function) that maybe used to test whether or not the above extractWords function works.	<pre> int main() { char mySent, * strcpy, * myStr, exwdc[MAXLEN]; int nb words; extractWords(myStr, exwdc, &nb words); } </pre>

V. (16 pts) Workout Problems

For the following questions, you don't need to write the calling function (nor the main). All communication shall be made thru the return statement and/or thru parameter passing.

1. Declare and write the code for a function, named **deleteEvens**, that deletes all even numbers from a given array of numbers.

```
#include <stdio.h>
using namespace std;
void deleteEvens(double even[]);
int numbers;
number = even[]
if (number % 2 == 0) {
    number = 0
}
```

2. Declare and write the code for a function, named **insertElt**, that inserts a number in an already sorted array of numbers (in ascending order). It is assumed that the sorted array will be transferred from the calling function.

```
#include <stdio.h>
using namespace std;
void sort(float a, int n) {
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (a[i] > a[j])
                swap(a[i], a[j]);
}
void sort(float a, int n) {
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (a[i] > a[j])
                swap(a[i], a[j]);
}
```

```
int main() {
    int n, p;
    float b[];
    sort(b, n);
    insertElt(b, n, p);
    return 0;
}
```

25. (5 pts) A module, named **identicalList** that tells whether or not two lists of same size are identical. You may consider the list to contain integers.

Incoming: -----

Two list of integers, sorted & identical

Outgoing: -----

True/False (1/0)

Prototype: -----

bool identicalList(const int a[], const int b[], int size)

IV. (30 pts) Short Answer Questions 2.

All the questions in this part are based on the function **extractWords** that extracts and counts all the words that maybe found in a given line of text. For the sake of simplicity, the line of text is assumed to have no punctuation marks. Line numbers are added to facilitate the reading of the source code.

```

1. const SPC = ' ', // the SPACE character
2. const MAXNUM = 20; // maximum nb of words in a sentence
3. const MAXLEN = 30; // maximum length of each word
4. void extractWords(const char * mystr, char extWd[][MAXLEN], int & nbWords)
5. { char * mysent, * spcptr;
6.   int i;
7.   strcpy(mysent, mystr);
8.   strcpy(mysent, " "); // to extract the last word
9.   i = 0;
10.  while (strlen(mysent) != 0)
11.  { while ( strlen(mysent) && (mysent[0] == SPC) ) mysent++;
12.    spcptr = strchr(mysent, SPC);
13.    if (spcptr != NULL) {
14.      strcpy(extWd[i], mysent, spcptr - mysent);
15.      extWd[i][spcptr - mysent] = '\0';
16.      i++;
17.      mysent = spcptr + 1;
18.    }
19.  }
20.  nbWords = i;
21. }
```

1. Identify the local variables	<i>mysent spcptr</i>	
2. Identify the pre-defined string functions used	<i>strcpy strchr strlen</i>	
3. Which formal parameter is used to bring the line of text from the calling function	<i>const char *mystr</i>	

III. (20 pts) Short Answer Questions I.

For each of the modules below, describe the incoming and outgoing data and then provide a prototype. Do not write the code!

21. (2 pts) A module, named **swap** that swaps two real numbers.

Incoming: two numbers x & y, and a space holder variable temp.

Outgoing: two swapped real nos.

Prototype: void swap(double &x, double &y);

22. (3 pts) A module, named **addAllPrimes** that adds and returns the sum of all the prime numbers below a given natural number.

Incoming: list of natural nos. 1 to n of which, prime nos. are selected.

Outgoing: sum of all prime nos.

Prototype: long addAllPrimes (const int &n, int &sum, int &primelist, long &sum);

23. (5 pts) A module, named **suppressVowels** that suppresses (deletes) all the vowels found in a line of text.

Incoming: text, length of the text, search text to remove vowels, and return back of how many vowels are removed.

Outgoing: delete found vowels.

Prototype: void suppressVowels(char *text, int len, int &sk, int &found);

24. (5 pts) A module, named **highestFrequentLetter** that identifies the highest frequent letter in a given line of text.

Incoming: text, length of the text, no. of occurrence of letter, max occurrence identified.

Outgoing: highest + frequent letter in a given line of text.

Prototype: void highestFrequentLetter(char *text, int len, int &sk, int &max);

4.	Which formal parameter is used to transfer the extracted words to the calling function	char ext wd[1][MAXLEN]
5.	Which formal parameter is used to transfer the number of words information to the calling function	int & nWords ✓
6.	List the protocol(s) of communication used	P ₂ , P ₃ ✓
7.	List the tools of communication used	T ₃ ✓
8.	What is the purpose of line # 7	It copies ~~mystr~~ to mySent to avoid modification of the string(line of text).
9.	What is the purpose of the keyword const in the first formal parameter	
10.	Which line of code actually searches for the next word to extract	16 ✓
11.	(4 pts) If the number of words information is to be sent thru the function identifier (ie. T2), rewrite the declaration (only) of extractWords to reflect this change	int extractWords(const char *mystr, char ext wd[1][MAXLEN], int nWords);
12.	(6 pts) Write a function (could be the main function) that may be used to test whether or not the above extractWords function works.	<pre> int main() { const char *mystr; int i; char ext wd[1][50]; cout << "Enter your line of text:"; cin >> mystr; extractWords(mystr, ext wd[1][50], i); cout << "The extracted words are:" << i << endl; return 0; } </pre>