MAKERERE UNIVERSITY

COLLEGE OF ENGINEERING DESIGN, ART AND TECHNOLOGY
SCHOOL OF ENGINEERING
ELECTRICAL AND COMPUTER DEPARTMENT

CMP1201-COMPUTER PROGRAMMING FUNDAMENTALS

SEMESTER II, 2018/2019 FINAL EXAMINATION
THURSDAY, MAY 2ND 2019,8:00AM-NOON

INSTRUCTIONS:

a) This examination is composed of two sections, section A and Section B. Section A is compulsory and carries a total of 40marks.

b) Attempt any three (3) questions from Section B for a total of 60 marks.

c) All the language, terminology and references in this paper must be interpreted and understood in the context of c programming.

SECTION A: TRUE/FALSE (40 marks)

Answer “TRUE” or “FALSE” for each of the statements in Section A and provide a one statement reason for your choice of TRUE or FALSE for each statement. (2 points for each statement)

[1] Dynamically allocated memory using malloc() and calloc() only stays used/ available for the execution of the program.

[2] Passing elements of an array to a function is no different than passing a variable to a function except for the increased memory requirements.

[3] For the same number of data points, the insertion, bubble and selection sorts require the same time and storage complexity.

[4] Scanf() and gets() are applicable when manipulating strings and will always return the same results for the same sequence of character inputs.

[5] All programming operators in C are specified such that they can be applied to any data type as the programmer deems fit.

[6] Strings are defined as an array of characters and as such can be manipulated in the same way you would handle any other array.

[7] .txt and .bin files are equally as important in C programming because they contain system information.

[8] Operators precedence allows for the orderly evaluation of expressions and requires that those with the highest precedence be evaluated last.

[9] Pointers allow C programmers the ability access memory and manipulate addresses as seen fit, a feature that is often missing in other programming languages.

[10] In the ladder if ……. else statement, a TRUE evaluation of any of the if ……. else statement requires that all the statements after must be executed too.

[11] All the four types of loops in C programming allow the user to perform a function repeatedly with the same efficiency.

[12] One of the qualities of is good algorithm requires that the algorithms must provide the most effective way to solve a problem.

[13] When used in a program, structures and unions have the same memory requirements.

[14] Once memory is dynamically allocated, there is no way one can modify it irrespective of the program needs at hand.

[15] Global and local variables of a program can be accessed by any function within the program during execution.

[16] The storage class of a variables determines its data type and lifetime.

[17] User-defined functions are similar to standard library functions and follow the same basic syntax when used.

[18] Since operations on a stack can only be performed from one end only, stacks are effectively LIFO data structures.

[19] The float and double data types can be used to hold real numbers in a program. As a result, they require the exact same memory for storage.

[20] The use of bitwise operators allows for faster processing because the symbols used are more acceptable to the processors in the computing devices.

SECTION B: (60 marks)

Question 1:

a) What is a programming language? (1)

i. List two categories of programming languages and explain how they differ. (4)

b) The main () function in C provides the starting point in the execution of a program.

i. What is a function and under what categories are functions classified? (2)

ii. Discuss the benefits of classifying functions as described in b(i) above. (3)

c) The Fibonacci sequence is a series where the next number of the sequence is the sum of the previous two (2) numbers. The first two elements of Fibonacci series is 0 and 1 respectively. Write a program that computes and returns to the user the first 10 numbers in the Fibonacci series. (10)

Question 2:

a) Study the piece of code and answer the questions that follow:

```
 #include <stdio.h>
int main ()
{
  int marks [10], i, n, sum=0, average;
  printf ("Enter n: ");
  scanf ("%d", &n);
  for (i=0; i<n; ++i) {
    printf ("Enter number %d: “, i+1);
    scanf ("%d”, &marks[i]);
    sum += marks[i];
  }
  average = sum/n;
  printf ("Average = %d”, average);
  return 0;
}
```

i. What function does the above program serve? (1)

ii. Identify the variables in the program and explain the logic behind the variable choices made. (4)

iii. What is the main difference between a variable and a constant and hoe do the attributes of the and storage class apply to either of them? (3)

iv. List three types of variables and discuss how they are used. (3)

v. Modify the above program to process the marks for five different students each registered for 4 course units and assign a PASS or FAIL for a students each whose average score is at least 50% or less than 50% respectively. (9)

Question 3:

a) Sorting is one of the mechanism of processing data into an ordered format. Several sorting mechanisms exits for this purpose.

   i. Name three different sorting techniques available to a programmer and discuss the two main features that each technique required for proper operation. (4)

   ii. For the sorting mechanisms listed above, use an array of integers to provide a step-by-step illustration of the operation of each of the sorting mechanisms. (8)

   iii. Using one of the sorting mechanisms listed above, write a program that implements the sorting of an array of integers in ascending order. (8)

Question 4:

a) Explain the basic datatypes in C. (3)

b) What are the relational, logical and bitwise operators? (3)

c) Looping in C can be implemented in any of the three ways. List the three different looping techniques and explain the syntax of each one of them. (6)

d) For each of the following code extracts, determine their output. (1.5)

```
i.        int m=1, n=2;
          for (j=1; j<=2; j=j+1)
          {
           m=m+1;
           n=n*j;
           print ("%d \t %d\t”, m, n);
          }
```

```
ii.       main ()
        {
         int a [7] = {11,12,13,14,15,16,17};
          int i;
         printf ("content of the array");
        for (i=0; i<=6; ++i);
         {
        printf ("%d\t”, a[i]);}
        }
```

e) Rewrite each of the above pieces of code extract in e(i) and e(ii) above using the while loop (5)

END