UNIVERSITY OF RUHUNA
BACHELOR OF COMPUTER SCIENCE (GENERAL) DEGREE
LEVEL I (SEMESTER I) EXAMINATION – April 2023
URSE UNIT: CSC1113 (Programming Technique) – Theory **TIME: 2 HOURS**

Answer <u>All four</u> (04) Questions.

a. State whether the following rules for variable naming are True or False in C programs.
   i. Variable names **UserName** and **username** are identical.
   ii. Spaces between characters in the name are allowed.
   iii. The underscore character is the only special character that can appear in a variable name.
   iv. A variable name should not exceed 255 characters.
   v. The name must begin only with a letter in the English alphabet.

b.
   i. Write the ouput of the C program given in Figure 1.
   ii. Which conditional expression need to be changed to display the output as 15? Mention the required change in the conditional expression.
   iii. Write the output of the C program if the bold colour expression in Figure 1 changed as **(x+2*y<pow(z,2))**
   iv. What is the type of the ternary operator in C programs and list the three arguments it takes.

```
#include <stdio.h>

void main()
{
   int x=10,y =6;
   int z= 15;

   if(x != y)
   {
     if ((x < y) || (x <= z) || (x >= y+4) )
        printf("%d\n",x);
     if ((x >y ) && (y <=z))
        printf("%d\n",y);
   }
   else
   {
      printf("%d\n",z);
   }
}
```

Figure 1

c.

i. Write complete C program statements to achive following tasks.
   A. Declare single precision variable **_gpa** to hold GPA with decimal values.
   B. Declare an integer variable name **uid**.
   C. Initialise a character variable name **Opt** with first character in the English alphabet.
   D. Declare an interger constant with the name **MODYFIER** as value 10.
   E. Use macro definition to declare a constant **PI** with value 3.14.

ii. Which datatype should be used to have higher precision for the declaration in *1.c. i.A* above?

iii. What should be done to have more memory for declaration in *1.c.i.B* above?

d.

i. Fill in the blanks marked with label A to D in the C programs (Figure 2) to get the output of **10 12 14 16**.

```
#include <stdio.h>

void main()
{
   for(...A.....; i<18;...B....)
   {
      printf("%d ",i);
   }

}
```

```
#include <stdio.h>

void main()
{
   ...C.....;
   while(i<18)
   {
      printf("%d ",i);
      .....D....;
   }
}
```

Figure 2

ii. What is the format specifier used in the Figure 2 Programs? Explain the use of that format specifier for the ouput of programs.

a. State whether following statements are True or False for functions in C programs.
   i. Code encapsulation is a benefit of user defined functions.
   ii. Parameter *pass by value* approach in calling a function uses same memory for both actual and formal parameters.
   iii. A function header can only include return type and function name.
   iv. Built-in function **strcmp()** returns positive value if two strings are equal.
   v. Data type of the return value of function should be similar to return type in the function header.

b.
   i. Write the built-in functions that should use to achive the following tasks.
      A. Returns the amount of memory allocated to a data types.
      B. Returns the square root of floating point number.
      C. Converts every character to lowercase in a string.
      D. Print a simple text sentence or value of any variable.
      E. Returns the length of string "CSC1113".
      F. Read formatted input from standard input.

   ii. Briefly explain how recursive functions work in a program. Give an application of recursive functions in a program to solve a problem.

c. Figure 3 (a) and Figure 3 (b) show C programs written using **pass by values** and **pass by reference** approaches respectively.
Fill in the blanks marked with label A to E in the C programs to get the summation of two declared variables as output.

```
#include <stdio.h>
.....A.....

void main()
{
   int x=10;
   int y=20;
   int z=calAdd(x, y);
   printf("%d",z);
}

int calAdd(int a, int b)
{
   .....B.....
}
```

3a – pass by value

```
#include <stdio.h>
.....C.....

void main()
{
   int x=10;
   int y=20;
   int z=calAdd(&x, &y);
   printf("%d",z);
}

int calAdd(.....D.....)
{
   .....E.....
}
```

3b – pass by reference

Figure 3

d. Complete blanks labeled with 1-12 in the C Programs in Figure 4 using appropriate terms from the below list to get the output indicated.
(Note: You can use a word multiple times)

<math.h>, %d , %s, %f, char, strlen, strcmp, <stdlib.h>, float, %.1f, a, b , int, strcpy, strcat, %.lf, <string.h>

```
#include <stdio.h>
(1) ..... maxCal(..... (2) ..... a, ..... (3) ..... b)
{
  if(a>b)
    return ..... (4) .....;
  else
    return ..... (5) .....;
}

void main()
{
  float max=maxCal(10.2, 20.4);
  printf("..... (6) .....",max);
}
```

Output : 20.4

```
#include <stdio.h>
#include ..... (7) .....

void main()
{
  ..... (8) ..... str1[ ]="Programming";
  ..... (9) ..... str2[ ]="Techniques";
  ..... (10) .....(str1,str2);
  printf("..... (11) .....\n",str1);
  printf("%d",..... (12) .....(str1));
}
```

Output : ProgrammingTechniques
21

Figure 4

3.

a. State whether following statements are True or False for arrays in C Pograms.
   i. An array needs to be declared with the size in advance.
   ii. An array can be initilised at the time of declaration.
   iii. The **sizeof()** buit-in function will give the number of elements of an array.
   iv. A multidimensional array can only represent 2 or 3 dimensions.
   v. A string can be defined as a character array.

b. Table 1 summarises marks of five students for three subjects.

Student ID	*Subject Details*		
	Database	*C Programming*	*HTML*
0001	76	45	67
0002	60	65	62
0003	56	75	65
0004	78	65	72
0005	87	90	80

Table 1

i. Write the most suitable type of array to store the above details appropriately using a C Program with the number of dimensions.

ii. Write a C program to create an array to store the above details and print the marks for each student in the array you created using loop control structures to output Student ID and marks. [Note: Expected Output of the program is given in Figure 5]

```
Student ID 0001 marks are: 76 45 67
Student ID 0002 marks are: 60 65 62
Student ID 0003 marks are: 56 75 65
Student ID 0004 marks are: 78 65 72
Student ID 0005 marks are: 87 90 80
```

Figure 5

iii. Assume that, course coordinator needs to change the marks for *C Programming* for the student with ID **0003** as 70. Write a C Program statement to perform this task.

c. Fill in the blanks in below statements by selecting the most appropriate term within the brackets.

i. A incremented pointer point to the …… memory location. **[next, previous, first]**

ii. A pointer is a variable whose value is the …… of another variable. **[name, address, value]**

iii. The main benefit of using pointers is to write …….. programs. **[scalable , memory efficient, short]**

iv. A pointer can also be used to refer to ……. **[pointer, program, class]**

v. The value of pointed variable can be obtain using ..... Operator **[dereference, ternary, sizeof]**

d. Briefly describe following terms related to pointers.
   i. void pointers
   ii. malloc function

e. Write the outputs produced by the statements label with A-G characters in following C programs with pointers.

i.

```
#include <stdio.h>

void main()
{
   int num1=10;
   int num2=5;
   int *ptr =&num1;
   printf("%d",*ptr); //Output: Lable A
   *ptr= 8;
   printf("%d",*ptr+num2); //Output: Lable B
}
```

ii.

```
#include <stdio.h>

int main()
{
    int a=10,b=5;
    int *pn1, *pn2;
    int temp;
    pn1=&a, pn2=&b;
    printf("%d,%d\n",*pn1,*pn2); //Output: Lable C
    temp=*pn1;
    *pn1=*pn2;
    *pn2=temp;
    printf("%d,%d\n",*pn1, *pn2); //Output: Lable D
    printf("%d,%d\n",a, b); //Output: Lable E
}
```

iii.

```
#include<stdio.h>
void main()
{
    int a[5]= {12,3,4,7,5,8};
    int *s,i,small;
    s=&a;
    small=*s;
    printf("Smallest Element : %d\n",small); //Output: Lable F
    for(i=0;i<5;i++,s++)
        if(*s<small)
            small=*s;
    printf("Smallest Element : %d",small); //Output: Lable G
}
```

a. How does a structure in *C* language differ from an array?

b. Explain the following terms giving an example for each of them.
   i. Nested Structures
   ii. Array of Structures

c. Write the outputs of the following *C* programs.

i.

```
#include <stdio.h>
struct result{
  char sub[20];
  int marks;
};
void main()
{
    struct result res[] = { {"Maths",100},
    {"Science",90},
    {"English",85}
    };
   printf("%s ", res[1].sub);
   printf("%d", (*(res+2)).marks);
}
```

ii.

```
#include <stdio.h>
struct abc{int a; int b;} v[3], *p;
int     main()
        {
                p=v;
                p->a=3;
                p->b=p->a;
                printf("\n%d\t%d", v[0].a, v[0].b);
                return 0;
        }
```

d.

i. Define a structure called *cricket* that will describe the following information on cricket teams and players; Player name, Team name and Batting average. Assume that the player name is 30 characters long, team name contains 20 characters and batting average is an integer.

ii. Using *cricket* structure, declare an array named player with 100 elements and write a C code segment to read the information about all the 100 players and print team wise list containing names of players with their batting average.

e.

i. File access in C is normally sequential. However, a file can be read or written in any order, using standard C functions available. Name such a function and explain the usage of it giving all the parameters.

ii. Write a C program to write 10 integers from 1 to 10 into a file called "mydata".

********** END **********