VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Slot: A2

School of Computer Science Engineering and Information Systems

Winter Semester 2023-2024

Continuous Assessment Test – I

Programme Name & Branch B.Tech. IT

Course Name & Code: BCSE102L Structured and Object Oriented Programming

Class Number (s): VL2023240503278, VL2023240503275, VL2023240503263

Faculty Name (s) Dr. Chandra Mouliswaran S, Dr. Thanapal P, Dr. Rampriya R S

Exam Duration: 90 Min. **Maximum Marks: 50**

Answer Key

Q.No.	Questions	Max Marks	CO	BL
1.	Create a C program to print the Pascal's triangle and invert the Pascal's triangle by using 'nested while loops' only. The program will take number of rows as input from users. Then use 'nested while loops' to print the Pascal's triangle. The output of the program should be as in the following pattern if you enter the number of rows =5. **Test Case:** Enter the number of rows: 5 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 1 3 3 1 1 2 1 1 1 1 **Answer**	10	CO 1	BL 3

```c
#include<stdio.h>
void main(){
int rows,coef=1,space,i,j;
printf("Enter number of rows: ");
scanf("%d",&rows);
i=0;
while(i<rows){
    space=1;
    while(space<=rows-i){
        printf("  ");
        space++;
    }
    j=0;
    while(j<=i) {
        if (j==0||i==0)
            coef=1;
        else
            coef=coef*(i-j+1)/j;
    printf("%4d",coef);
    j++;
    }
    printf("\n");
    i++;
}
// Invert Pascal's triangle
i=rows-2;
while(i>=0){
    space=0;
    while(space<rows-i){
        printf("  ");
        space++;
    }
    j=0;
```

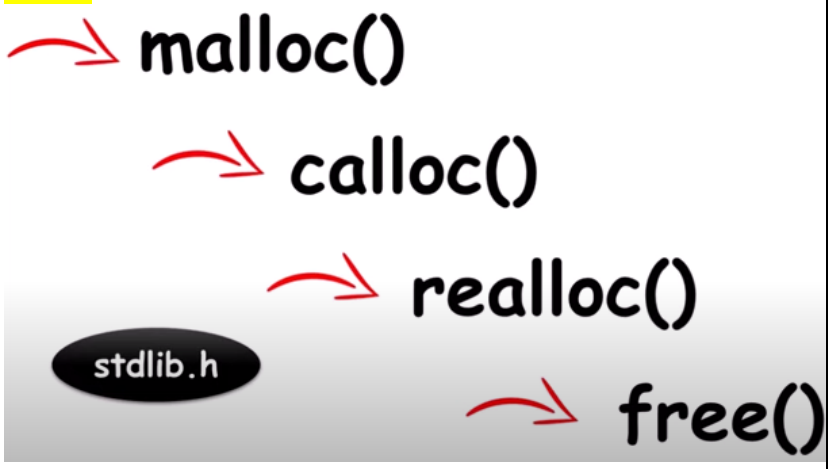
```
        while(j<=i) {
            if (j==0||i==0)
                coef=1;
            else
                coef=coef*(i-j+1)/j;
        printf("%4d",coef);
        j++;
        }
        printf("\n");
        i--;
    }
    }
```

```
Enter number of rows: 5
           1
         1   1
       1   2   1
     1   3   3   1
   1   4   6   4   1
     1   3   3   1
       1   2   1
         1   1
           1
```

Mark Distribution:
To initialize/get an input and variables declaration – 2m
Upper triangle code (3 nested while loops) – 4m
Lower triangle code (3 nested while loops) – 4m

2.	Write a C program to implement lift functions of an apartment. The user should be able to enter a destination floor between 0 and 5. The program has to iterate until user has reached the destination. Once the user has reached the destination, user confirms it by giving the option 1. **Test Case:** We have 5 floors, you can move from 0 to 5 *(Note: Press -1 if you reached your destination)* ---------- Enter the floor number where you want to go :- (0 to 5) 3 You are now on 3rd floor Enter the floor number where you want to go :- (0 to 5) 1 Hope you reached your destination! Visit Again! **Answer**	10	CO 1	BL 3

```c
#include <stdio.h>

int main() {

    int currentFloor = 1;
    int destinationFloor;

    printf("We have 5 floors, you can move from 0 to 5\n");

    do {
        printf("-------------------------------------------------------\n");
        printf("Enter floor no. where you want to go (0 to 5):- ");
        scanf("%d", &destinationFloor);

        if (destinationFloor > currentFloor||destinationFloor==0) {
            printf("You are now on floor %d\n", destinationFloor);
        } else {
            printf("Hope you reached your destination! Visit Again!\n");
        }

    } while (destinationFloor != currentFloor);

    return 0;
}
```

	`` We have 5 floors, you can move from 0 to 5 -- Enter floor no. where you want to go (0 to 5):- 3 You are now on floor 3 -- Enter floor no. where you want to go (0 to 5):- 0 You are now on floor 0 -- Enter floor no. where you want to go (0 to 5):- 4 You are now on floor 4 -- Enter floor no. where you want to go (0 to 5):- 1 Hope you reached your destination! Visit Again! ``` **Mark Distribution:** **To get an input and variables declaration – 2m** **do-while/equivalent loop statement with condition – 4m** **if-else /equivalent statement with condition – 4m**			
3.	List various functions provided by C language to facilitate dynamic memory allocation and explain each in detail with examples. **Answer**  • Should explain above functions with syntax and examples **Mark Distribution:** **malloc() definition, syntax, example – 3 m** **calloc() definition, syntax, example – 3 m** **realloc() definition, syntax, example – 3 m** **free() definition, syntax, example – 1 m**	10	CO 2	BL 2
4.	a) Code a C program which checks whether the entered substring is present in the input string. **[6 M]** **Test Cases:** Enter a string: hello Enter search substring: world SEARCH UNSUCCESSFUL! Enter a string: helloworld Enter search substring: ld SEARCH SUCCESSFUL! **Answer**	10	CO 1	BL 3

```c
#include<stdio.h>
int main(){
    char str[80], search[10];
    int count1 = 0, count2 = 0, i, j, flag;
    printf("Enter a string:");
    gets(str);
    printf("Enter search substring:");
    gets(search);
    while (str[count1] != '\0')
        count1++;
    while (search[count2] != '\0')
        count2++;
    for (i = 0; i <= count1 - count2; i++){
        for (j = i; j < i + count2; j++){
            flag = 1;
            if (str[j] != search[j - i]){
                flag = 0;
                break;
            }
        }
        if (flag == 1)
            break;
    }
    if (flag == 1)
        printf("SEARCH SUCCESSFUL!");
    else
        printf("SEARCH UNSUCCESSFUL!");

    return 0;
}
```

```
Enter a string:hello
Enter search substring:world
SEARCH UNSUCCESSFUL!
```

```
Enter a string:helloworld
Enter search substring:ld
SEARCH SUCCESSFUL!
```

Program Explanation

In this C program, we are reading a string using gets() function 'str' character variable. We are reading value another string to search using search character variable. To check substring is present in the given string. While loop is used to compute the str[] and search[] array variable value is not equal to null.

If the condition is true then execute the iteration of the loop. Increment the values of 'count1 and count2 variable values. Now we are using two for loops to check if the substring is present in the given string. We are initializing the 'i' variable value to 0 and the loop will execute till the condition that 'i' variable value should be less than or equal to the difference of count1 and count2 variable values.

If the condition is true, then another for loop will execute by initializing the 'i' variable value to 'j' variable. And the loop will terminate if the 'j' variable value is less than the sum of 'i' variable value with count2 variable value if the condition is true.

Then it will execute the loop by assigning the flag variable value as 1 and if condition statement is used to check the str[] array variable value is not equal to search[] with the base index is the difference between 'j' variable and 'i' variable value. If the condition is true then it will execute the statement and assign flag variable value as 0. For loop iteration will terminate till the condition becomes false. If-else condition statement is used to check if flag variable value is equal to 1 then print as search successful otherwise if the condition is false then print the statement as search unsuccessful.

Mark Distribution:

To initialize/get an input and variables declaration – 2m

Search Logic implementation using necessary branching and looping statements –4m

b) Write the output for the following C program. **[4 M]**

```
#include <stdio.h>
void main()
{
int a[5]={5,1,15,20,25};
int i,j,k=1,m;
i=++a[1];
j=a[1]++;
m=a[i++];
printf("\n%d%d%d",i,j,m);
printf("\n%d",a[1]);
}
```

Answer

```
3215
3
```

Mark Distribution:
3215 – 3m
3 – 1m

5.	Difference between the following with example each: a) Call by Value vs. Call by Reference	10	CO 2	BL 2

S. No.	Call by Value	Call by Reference
1.	A copy of actual parameters is passed into formal parameters.	Reference of actual parameters is passed into formal parameters.
2.	Changes in formal parameters will not result in changes in actual parameters.	Changes in formal parameters will result in changes in actual parameters.
3.	Separate memory location is allocated for actual and formal parameters.	Same memory location is allocated for actual and formal parameters.

b) Entry Control Loop vs. Exit Control Loop

Entry Controlled	Exit Controlled
Condition is checked at the entry of the loop	Condition is checked at the exit of the loop
If condition is initially false, the loop never executes **i=1;** **while(i==0)** **{** **System.out.println("In While loop");** **}** **System.out.println("out of the loop");** Output: **Out of the loop**	If condition is initially false, then also the loop executes at least once **i=1;** **do** **{** **System.out.println("In While loop");** **} while(i==0);** **System.out.println("out of the loop");** Output: **In while loop** **Out of the loop**
Example- for, while	Example – do-while

c) Predefined Function vs. User-defined Function

Library Function	User Defined Function
(1)A function which is already defined in compiler is called library function. Exp-scanf(),clrscr()	(1) A function which is created by user for its own programming requirement is called user defined function.Exp-main()
(2) We don't need to write a code for library function.	(2) But in user define function we have to write a code.
(3) The name of library function can't be changed because its prototype predefined in the compiler.	(3)The name of user define function can be changed.

d) Looping Statement vs. Branching Statement

Branching	Looping
Branching statements alter the flow of the program based on conditions.	Looping statements allow for repetitive execution of code based on a specified condition.
Branching statements, such as if-else and switch-case, allow the program to execute different blocks of code based on certain conditions. These conditions are evaluated and the program branches to different sections of code accordingly.	Looping statements, such as for, while, and do-while, allow the program to execute a block of code repeatedly based on a specified condition. This helps in automating repetitive tasks and iterating over data structures.

e) Pointer vs. Array

POINTERS	ARRAY
Pointer is a variable which stores the address of another variable.	Array is a collectihon of homogeneous data elements.
Pointer can't be initialized at definition.	Arrays can be initialized at definition.
They are static in nature.	They are static in nature.
The assembly code of pointer is different that array.	The assembly code of an array is different than pointer.

Mark Distribution:
Each subdivision carries 2 m with min of 2 points each.

NOTE*: Please refer below to the BL – Bloom's Taxonomy Levels and mention the respective level in the questions.

Bloom's Taxonomy Levels	Category
BL1	Remembering
BL2	Understanding
BL3	Applying
BL4	Analyzing
BL5	Evaluating
BL6	Creating