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

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CAT I , February 2024

B. Tech -Winter Semester 2023_2024

Slot:C1

Course Code : BCSE102L **Duration** : 90 Minutes

Course Title : Structured and Object Oriented Programming **Max. Marks** : 50

Answer all the questions

Q.NO	QUESTION

1	Consider an application that allows the user to enter 5 integer values, which includes both positive and negative numbers. The application will display the counts the number of positive numbers and negative numbers in the application. Use C Programming.
	int main() {
	float numbers[5];
	int j, pctr=0, nctr=0;
	printf("\nInput the first number: ");
	scanf("%f", &numbers[0]);
	printf("\nInput the second number: ");
	scanf("%f", &numbers[1]);
	printf("\nInput the third number: ");
	scanf("%f", &numbers[2]);
	printf("\nInput the fourth number: ");
	scanf("%f", &numbers[3]);
	printf("\nInput the fifth number: ");
	scanf("%f", &numbers[4]);
	
	for(j = 0; j < 5; j++) {
	if(numbers[j] > 0)
	{
	pctr++;
	}
	else if(numbers[j] < 0)
	{
	nctr++;
	}
	}
	printf("\nNumber of positive numbers: %d", pctr);
	printf("\nNumber of negative numbers: %d", nctr);
	printf("\n");
	return 0;
	}

2	Do you think Nested if statement and else if ladder statements are similar? Justify your answer with example programs. Nested if statements and else-if ladder statements are similar in that they both provide branching logic based on multiple conditions. However, they differ in terms of their structure and usage.

	Nested if statements: • Nested if statements are used when you need to check multiple conditions sequentially, where the execution of one condition depends on the outcome of another. • Each if statement is nested within another if statement, creating a hierarchical structure. • They provide flexibility in handling complex conditions but can lead to code indentation and readability issues if nested deeply. **Else-if ladder statements**: • Else-if ladder statements are used when you have multiple conditions to check, and you want to execute only one block of code based on the first condition that evaluates to true. • It consists of a series of if-else statements where each else-if statement is evaluated sequentially until one condition is found to be true. • They are more readable and structured for scenarios where each condition is mutually exclusive.
3	What are Storage Classes in C? List and illustrate all the types of Storage Classes with declaration syntax and example program for each? C Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility, and lifetime which help us to trace the existence of a particular variable during the runtime of a program. • **auto:** Variables declared with the auto storage class in C are automatically created within a code block (usually within a function) and are destroyed when the block exits. This storage class is default for all local variables in C, so explicitly using "auto" is optional. • **register:** The "register" storage class is used to suggest to the compiler that a variable be stored in a CPU register for faster access. However, it's merely a suggestion, and the compiler might or might not comply. Like "auto" variables, "register" variables have block scope. • **static:** Variables with the "static" storage class persist throughout the program's execution. A static variable retains its value between function calls when declared inside a function. If declared outside a function, it becomes accessible only within the file where it's declared, acting as a file-local variable. • **extern:** The "extern" storage class is used to declare variables or functions defined in other files. It's used to access global variables or functions defined in a separate file from where they're being used. When used to declare a variable, it refers to a global variable defined elsewhere. **Syntax** **storage_class** var_data_type *var_name*; examples for each
4	Imagine you are tasked with organizing a programming competition where participants need to create a C program. The challenge involves creating two arrays, each with a minimum of 4 elements. The first array should remain unchanged, while the second array needs to be reversed. Participants must then merge these two arrays, sort the resulting array, and finally, print the sorted array. Implement the solution by passing the arrays to appropriate functions. Sample Input: Enter the number of elements for First Array : 4 Enter the elements for First Array : 4 13 12 1 Enter the number of elements for Second Array : 4 Enter the elements for Second Array : 6 7 8 9 Output: Elements After Merging 4 13 12 1 9 8 7 6 The sorted elements are 1 4 6 7 8 9 12 13

```
int main()
{
	int ar1[40],temp,i,n1,n2;
	printf("Enter the no. of elements for both arrays: ");
	scanf("%d%d",&n1,&n2);
	if(n1>=4 && n2>=4)
	{
		//Taking Input for both the Arrays:
		printf("Enter the elements of first array: ");
		for(i=0; i<n1; i++)
		{
			scanf("%d",&ar1[i]);
		}
		printf("\n Enter the elements of Second Array: ");
		int ar2[n2];
		for(i=0; i<n2; i++)
		{
			scanf("%d",&ar2[i]);
		}

		//Reversing the Second Array
		 int c=n2-1;
		for(i=0; i<n2/2;i++)
		{
			temp=ar2[i];
			ar2[i]=ar2[c];
			ar2[c]=temp;
			c--;
		}
		//Merging two arrays
		int ar3[100];
		int n3=n1+n2;
		for(i=0; i<n1; i++)
		{
			ar3[i]=ar1[i];
		}
		int j=0;
		for(i=0; i<n2; i++)
		{
			ar3[i + n1]=ar2[i];
		}
		printf("Elements after merging:\n");
		for(i=0; i<n3; i++)
		{
			printf("%d ",ar3[i]);
		}
		for(i=0; i<n3; i++)
		{
			for(j=i+1; j<n3; j++)
			{
				if(ar3[i]>ar3[j])
				{
					temp=ar3[i];
					ar3[i]=ar3[j];
					ar3[j]=temp;
				}
			}
		}
```

```
                printf("\n Sorted Array:\n");
                for(i=0; i<n3; i++)
                {
                        printf("%d ",ar3[i]);
                }
        }
        else
        {
                printf("Invalid Input !");
        }
        return 0;
}
```

5

a. Develop a C program for a sports analytics application. The application stores the performance scores of athletes in an array. You need to write a function “Score” that takes as input an array of integers representing the athletes' scores and the size of the array. The function “Score” should return a pointer to the maximum score in the array.

```
#include <stdio.h>

// Function to find the maximum score in the array
int* Score(int scores[], int size) {
   if (size == 0) {
      return NULL; // If the array is empty, return NULL
   }

   int maxIndex = 0; // Initialize index of the maximum score to 0

   // Loop through the array to find the index of the maximum score
   for (int i = 1; i < size; i++) {
      if (scores[i] > scores[maxIndex]) {
         maxIndex = i; // Update the index of the maximum score
      }
   }

   // Return a pointer to the maximum score in the array
   return &scores[maxIndex];
}

int main() {
   int athleteScores[] = {85, 92, 78, 95, 88};
   int size = sizeof(athleteScores) / sizeof(athleteScores[0]);

   // Call the Score function to find the maximum score
   int* maxScore = Score(athleteScores, size);

   // Check if maxScore is not NULL before dereferencing it
   if (maxScore != NULL) {
      printf("The maximum score is: %d\n", *maxScore);
   } else {
      printf("No scores found.\n");
   }

   return 0;
}
```

b. Develop a C program to enable the swapping of two numbers. The objective is to create a function “swap” that utilizes a temporary variable and pointer-based approach for exchanging the values of two given integers.

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

```
void swap(int *a, int *b) {
    int temp = *a; // Store the value of 'a' in a temporary variable
    *a = *b;       // Assign the value of 'b' to 'a'
    *b = temp;     // Assign the value stored in the temporary variable to 'b'
}

int main() {
    int x = 5, y = 10;

    printf("Before swapping: x = %d, y = %d\n", x, y);

    // Call the swap function to swap the values of x and y
    swap(&x, &y);

    printf("After swapping: x = %d, y = %d\n", x, y);

    return 0;
}
```