SET-A

(171)

Double side (8)

Scoring Indicators

Course Name : Programming in C
Course Code : (2021)-3132 **QID : 2110220122.**

Qn. No.	Scoring Indicator	Split score	Sub Total	Total score
I	**PART - A**			9

1	for (initialization Statement; test Expression; update Statement)
	{
	// statements inside the body of loop
	}
		1	1	

2	#define, #include, #undef…..	0.5+0.5	1	
3	\0 (NULL)	1	1	
4	<data_type> <array_name> [array_size]={list of elements}; eg. int a[] = {10, 20, 30, 40, 50};	1	1	
5	datatype *pointer_variableName;	1	1	
6	void free(void *ptr)	1	1	
7	'a'	1	1	
8	Structure in c is a user-defined data type that enables us to store the collection of different data types.	1	1	
9	Text Files, Binary Files	1	1	
II	**PART - B**			24
1	1.While the loop is an entry control loop because firstly, the condition is checked, then the loop's body is executed. The do-while loop is an exit control loop because in this, first of all, the body of the loop is executed then the condition is checked true or false. 2.The statement of while loop may not be executed at all. The statement of the do-while loop must be executed at least once. (Any two difference)	1.5 + 1.5	3	
2	Extern stands for external storage class. Extern storage class is used when we have global functions or variables which are shared between two or more files. Keyword **extern** is used to declaring a global variable or function in another file to provide the reference of variable or function which have been already defined in the original file. The variables defined using an extern keyword are called as global variables. These variables are accessible throughout the program. Notice that the extern variable cannot be initialized it has already been defined in the original file.	3	3	
3	Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column. Each element in the 2D array must by the same type. Syntax : data_type array_name[size][size];	2point (2)+syntax (1)	3	
4	The result is = : 0		3	
5	• Pointers save memory space. • Pointers are helpful in allocation and de-allocation of memory during the execution of the program • Pointers enhance the execution speed of a program. • Pointers are helpful in traversing through arrays and character strings. • Pointers reduce the execution time of programs …	Any 3x1	3	

6	int arr[5] = {3, 5, 7, 9, 11}, i;
	int *ptr = arr;
	
	// printing the elements of array using addition arithmetic on pointer
	for(i = 0; i < 5; i++)
	{
	printf("%d ", *(ptr + i));
	}
		Declaration and address assignment (1) + logic (2)
		
		** pointer can also be used for variable [illegible]
			3	

7	• Array elements are of the same data type while a structure has elements of different data types. • Array elements are stored in contiguous memory location. Structure elements may not be stored in a contiguous memory location. • Array elements are accessed by its position or subscript. Structure elements are accessed by its object as '.' operator.	Any 3 x 1	3	
8	These are used by the programmers to create their own data types and define what values the variables of these data types can hold. The keyword used to declare enumerated type is **enum**. **Syntax** enum tagname{ identifier1, identifier2,.....,identifier n }; **Example** enum week{ mon, tue, wed, thu, fri, sat, sun };	Statement (1) + syntax (1) + example (1)	3	
9	r- Read Only reading possible. No, create the file if it does not exist. w- WriteOnly writing is possible. Create the file if it does not exist; otherwise, erase the old content of the file and open a blank file. a- Append Only writing is possible. Create a file; if it does not exist, otherwise open the file and write from the end of the file. (Do not erase the old content). r+ - Reading + Writing Reading and writing are possible. Create a file if it does not exist, overwriting existing data. Used for modifying content. w+ - Reading + Writing Reading and writing are possible. Create a file if it does not exist. Erase old content. a+ - Reading + Appending Reading and writing are possible. Create a file if it does not exist. Append content at the end of the file.	Any 3x1	3	
10	Syntax - int main(int argc, char *argv[]) The arguments passed from command line are called command line arguments. These arguments are handled by main() function. When a program starts execution without user interaction, command-line arguments are used to pass values or files to it	Syntax(1)+any two point(2)	3	
III	**PART - C**			42
1	**a)** **1 2 3 4 5**	3	7	

	b)
	#include <stdio.h>
	int addNumbers(int a, int b); // function prototype
	
	int main()
	{
	int n1,n2,sum;
	
	printf("Enters two numbers: ");
	scanf("%d %d",&n1,&n2);
	
	sum = addNumbers(n1, n2); // function call
		Function declaration (1) + function calling in main() function (1) + function definition and logic (2)		

	return 0;
	}
	
	int addNumbers(int a, int b) // function definition
	{
	int result, avg;
	result = a+b;
	avg =result/2;
	printf ("The sum = %d",result);
	printf("the average = %d",avg);
	return 0; // return statement
	}			

	OR			
2	**Automatic, Register, Static, External** Automatic Scope:-local to the block or function in which variable is defined. Life time:-Till the control remains within function or block in which it is defined. It terminates when function is released Register Scope :-local to the function or block in which it is defined. Life time :-till controls remains within function or blocks in which it is defined. Static Scope :- local to the block or function in which it is defined. Life time:- value of the variable persist or remain between different function call (as long as program execution remains it retains) External Scope :- global Life time:-as long as program execution remains it retains	Listing (1) + 4 x 1.5	7	

3	A ***function*** is a group of statements that together perform a task.
	
	Every ***C*** program has at least one ***function***, which is main (),
	
	A user-defined function has three main components that are
	• function declarations,
	• function definition and
	• function call.
	
	A function **declaration/function prototype** tells the compiler about a
	
	function's name, return type, and parameters.
	
	A function **definition** provides the actual body of the function.
	
	Function definition
	
	return_type function_name(parameter list)
	
	{
	
	body of the function
	
	}
	Return Type – A function may return a value. The **return_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword **void**.
		Define Function (1) +definition (1x3) +example 3	7	

	OR			

4	**Recursion method**
	#include <stdio.h>
	int power(int n1, int n2);
	int main() {
	int base, a, result;
	printf("Enter base number: ");
	scanf("%d", &base);
	printf("Enter power number(positive integer): ");
	scanf("%d", &a);
	result = power(base, a);
	printf("%d^%d = %d", base, a, result);
	return 0;
	}
	
	int power(int base, int a) {
	if (a != 0)
	return (base * power(base, a - 1));
	else
	return 1;
	}
	
	Iterative method
	#include <stdio.h>
	int main() {
	int base, exp;
	long double result = 1.0;
	printf("Enter a base number: ");
	scanf("%d", &base);
	printf("Enter an exponent: ");
	scanf("%d", &exp);
	
	while (exp != 0) {
	result *= base;
	--exp;
	}
	printf("Answer = %.0Lf", result);
	return 0;
	}
		Recursion method (3.5) + Iterative method (3.5)	7	

5	#include <stdio.h>
	
	int main()
	
	{
	
	int a[10][10], sum=0,i,j,m,n;
	
	printf("Enter the rows and Column of first matrix\n");
	
	scanf("%d%d",&m,&n);
	
	printf("Enter elements of matrix\n");
	
	for (i = 0; i < m; ++i)
	
	for (j = 0; j < n; ++j)
	
	{
	
	scanf("%d", &a[i][j]);
	
	}
		Array and variable declaration (1) +Matrix reading (2.5) + sum calculation (2.5) +print sum (1)	7	

```
if(m==n)
{
   for (i = 0; i < m; ++i)
   {
     sum = sum+a[i][i];
   }
  printf( " The sum of Diagonal element of matrix is = %d",sum);
}
else
printf("Enter square matrix");
  return 0;
}
```

OR

6

```
#include <stdio.h>
int main()
{
 int array[100], n, c, d, position, t;
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
          for (c = 0; c < n; c++)
          scanf("%d", &array[c]);
                  for (c = 0; c < (n - 1); c++) // finding minimum
element (n-1) times
                  {
                   position = c;
                           for (d = c + 1; d < n; d++)
                           {
                            if (array[position] > array[d])
                            position = d;
                           }
                                    if (position != c)
                                    {
                                    t = array[c];
                                    array[c] =  array[position];
                                    array[position] = t;
                                    }
                  }
 printf("Sorted list in ascending order:\n");
   for (c = 0; c < n; c++)
   printf("%d\n", array[c]);
 return 0;
}
```

Variable, Array declaration (1) + reading values (1) + sorting logic (4) + print (1)

7

7

1) strlen(string_name)
returns the length of string name.
2)strcpy(destination, source)
copies the contents of source string to destination string.
3)strcat(first_string, second_string)
concats or joins first string with second string. The result of the string is stored in first string.

Listing(2)+ explanation (any 5 x1)

7

	4)strcmp(first_string, second_string) compares the first string with second string. If both strings are same, it returns 0. 5)strrev(string) returns reverse string. 6)strlwr(string) returns string characters in lowercase. 7)strupr(string) returns string characters in uppercase.			

8	#include <stdio.h>
	
	int main()
	{
	char str[100];
	int countL, countU;
	int counter;
	
	//assign all counters to zero
	countL = countU = 0;
	
	printf("Enter a string: ");
	gets(str);
	
	for (counter = 0; str[counter] != NULL; counter++) {
	
	if (str[counter] >= 'A' && str[counter] <= 'Z')
	countU++;
	else if (str[counter] >= 'a' && str[counter] <= 'z')
	countL++;
	}
	
	printf("Total Upper case characters: %d, Lower Case characters: %d", countU, countL);
	
	return 0;
	}
		Variable, Array declaration (1) + reading string (1) + Upper case and Lower case checking logic (4) + print (1)	7	

	OR			

9	The process of allocating memory at the time of execution or at the runtime, is called dynamic memory location. Allocation and release of memory space can be done with the help of some library function called dynamic memory allocation function.
	malloc() - This function use to allocate memory during run time, its declaration is
	void *malloc(size);
	malloc () - returns the pointer to the 1st byte and allocate memory, and its return type is void, which can be type cast such as:
	int *p=(datatype*)malloc(size);
	
	calloc() - Similar to malloc() only difference is that calloc() function use to allocate multiple block of memory. Two arguments are there, 1st argument specify number of blocks, 2nd argument specify size of each block.
	syntax:-
	int *p= (int*) calloc(number of blocks, size of each block);
	
	realloc() - The function realloc() use to change the size of the memory block and it alter the size of the memory block without loosing the old data, it is called reallocation of memory.
	It takes two argument such as;
		Statement (1) + any 3x2 (6)	7	

	int*p=(int *)realloc(ptr, new size); **free() -** Function free() is used to release space allocated dynamically. The memory released by free() is made available again. It can be used for further purpose. Syntax : free(p); , where p is pointer.			
	OR			

10	#include<stdio.h>
	#include<stdlib.h>
	int main(){
	int n,i,sum=0;
	int *p;
	//Reading number of elements from user//
	printf("Enter the number of elements : ");
	scanf("%d",&n);
	
	//Calling malloc() function//
	p=(int *)malloc(n*sizeof(int));
	
	if (p==NULL){
	printf("Memory not available");
	exit(0);
	}
	//Printing elements//
	printf("Enter the elements : ");
	for(i=0;i<n;i++)
	{
	scanf("%d",p+i);
	sum=sum+*(p+i);
	}
	printf("The sum of element is = %d", sum);
	return 0;
	}
		Variable, pointer declaration (1) + reading string (1) +using malloc function(2)+ calculate sum logic (2) + print (1)	7	

11	#include <stdio.h>
	/*structure declaration*/
	struct employee{
	char name[30];
	int age;
	char des[30];
	float salary;
	};
	
	int main()
	{
	int n,i;
	/*declare structure variable*/
	struct employee emp;
	printf("Enter how many employeee");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
	
	/*read employee details*/
	printf("\nEnter details :\n");
	printf("Enter the Name ?:\n");
	scanf("%s",emp.name);
	printf("Enter the Age:");
	scanf("%d",&emp.age);
		Structure definition (2) + structure variable declaration (1) + reading (2) + printing (2)	7	

```
    printf("Enter the Designation");
            scanf("%s",emp.des);
    printf("Salary ?:");
           scanf("%f",&emp.salary);
    }
    /*print employee details*/
     printf("\nEntered detail is:\n");
     for(i=0;i<n;i++)
     {

    printf("Employee Name: %s",emp.name);
    printf("Employee age: %d" ,emp.age);
    printf("Employee designation %s",emp.des);
    printf("Employee Salary: %f\n",emp.salary);
     }
    return 0;
}
```

OR

12 File is a collection of data that stored on secondary memory like haddisk of a computer. C programming language supports two types of files and they are as follows...

Text Files (or) ASCII Files

Binary Files

1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file
4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

File definition (1)+ listing (1)+exp(1x5) — 7

```
FILE *fopen( char *filename, char * mode );
      fp = fopen("file.txt", "r");
fclose(filepointername);
      fclose(fp);
int fprintf(FILE *stream, const char *format [, argument, ...])
      fprintf(fptr, "Id= %d\n", id);
int fscanf(FILE *stream, const char *format [, argument, ...])
      fscanf(fp, "%s %s %s %d", str1, str2, str3, &year);
int fputc(int c, FILE *stream)
       fputc('a',fp);
int fgetc(FILE *stream)
        while((c=fgetc(fp))!=EOF)
```