289

(D) AK. 1 Page: 12 April: 2024

Scoring Indicators

Course Name: Problem Solving and Programming
Course code: 2131

QID: 2103230002

Q NO	Scoring Indicators	Split score	Subtota	Total score
	PART-A			9
1	Flowchart	1	1	
2	printf()	1	1	
3	( ),*,<=,&&	1	1	
4	sizeof()	1	1	
5	Break	1	1	
6	Functions	1	1	
7	True	1	1	
8	19	1	1	
9	Array of arrays to store homogeneous data in tabular form	1	1	

	PART-B	Split score		24
1	Flowchart to calculate area(l*b) Flowchart to calculate perimeter2*(l+b)	**1** .5each		
2	Logical AND,OR,NOT with exact truth table	1each		
3	a) y=10 b) y=11	**1.5 each**		

4	Syntax Condition ? Statement1:Statement 2 If condition is true statement 1 will be executed else statement2 will be executed Example	1.5each		
5	Break is used to exit from a loop And next instruction in the program will be executed Continue is used to skip one single iteration of a loop And continues with the next iteration of the loop Difference-2marks Example-1/2 mark each			

6	i=1;
	while(i<=10)
	{
	printf("%d",i);
	i++;
	}
		3marks		

7	sqrt()-Math library to find square root of a number in math.h isdigit()-to check whether number isalpha()-to check whether alphabet islower()-to check whether lowercase pow()-Math function to compute power (Any 3 1 mark each)	1		
8	• In the Call by value method original value is not modified, whereas in the Call by reference method, the original value is modified. • In Call by value, a copy of the variable is passed, whereas in Call by reference, a variable itself is passed. • In Call by value, actual and formal arguments will be created in different memory locations, whereas in Call by reference, actual and formal arguments will be created in the same memory location.	1each		

9	Array-1mark
	
	Program Code
	
	#include <stdio.h>
	int main() {
	int n,max;
	int arr[50];
	printf("Enter the number of elements (1 to 100): ");
	scanf("%d", &n);
	for (int i = 0; i < 6; ++i) {
	printf("Enter number%d: ", i + 1);			

```
  scanf("%d", &arr[i]);
 }
max=arr[0];
 // storing the largest number to arr[0]
 for (int i = 0; i < 6; ++i) {
  if (max < arr[i]) {
   max = arr[i];
  }
 }

 printf("Highest mark = %d", max);

 return 0;
}
```

10 int arr[3][2]={{1,2},{2,3},{3,4},};
Declaration - 1.5mark
Initialization- 1.5mark

	PART C			56
III	It is a variable that stores data type value in a program.	It is similar to a variable and cannot be changed during program exec		
	It is a variable that can be changed after defining the variable in a program	It is a fixed variable that cannot be changed after defining the variabl program.		
	The value of a variable can change depending on the conditions.	In constants, the value cannot be changed.		
	Typically, it uses int, float, char, string, double, etc. data types in a program.	It can be express in two ways: #define pre-processor and t he const keyword.		
	Example: int a = 5; float radius = 5.2; char 'A';	**Example:** const int Len = 5; #define PI 3.14		
	Keyword is a pre-defined word.	The identifier is a user-defined word		
	It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.		
	Its meaning is pre-defined in	Its meaning is not defined in		

	the c compiler.	the c compiler.			
	It is a combination of alphabetical characters.	It is a combination of alphanumeric characters.			
	It does not contain the underscore character.	It can contain the underscore character.			
	Any 3 difference from each 1.5mak each				
	OR				
IV	Unary –Operator with single operand eg-,+,++,-- Binary-Operator with 2 operands +,-,*,/		1.5 each		
V	STRUCTURE(2marks) Include header file Global declaration Function main Declaration Execution Function call(optional) Comments(optional) Example(1mark)				
	OR				

VI	#include<stdio.h>
	
	int main()
	
	{
	
	int a=10, b=20;
	
	printf("Before swap a=%d b=%d",a,b);
	
	a=a+b;
	
	b=a-b;//
	
	a=a-b;//
	
	printf("\nAfter swap a=%d b=%d",a,b);
	
	return 0;
	
	}				

VII	Usage of scanf in exact format-2marks Condition if (num%2==0)even Else odd-2marks Overall correctness -3marks				

VIII	Switch statement is decision making statement multiple conditions With explanation-2marks switch(case variable) { case 1://statements break; case 2://statements break; case 3://statements break; . . . default://statements }(2marks) Example-3marks			

IX	<u>While loop</u>
	initialization expression;
	while(condition)
	{
	//body of loop
	}
	with example and explanation(3.5 marks)
	
	<u>do while</u>
	do
	{
	//body of loop
	}while(condition);
	with example and explanation(3.5 marks)			

	OR			
X	Using while loop or for loop Compute fact=fact*i until i is less than than1 Logic-2marks Exact Selection and syntax of loop-3marks Overall correctness-2marks			

XI	a) Function prototype includes
	• Return type
	• Function name
	• Parameter list
	• Body of function
	• Return value (2marks)
	Return-type function_name (parameter list)
	{
	//body
	return value; (2marks)
	}
	b)
	int sum_dig(int n)
	{
	
	int x=0;
	while(n>0)
	{
	y=n%10;
	x=y+x;
	n=n/10;
	}
	return x;
	
	} (2marks)
	
	Calling the function and displaying function in main program (1mark)
		N=	

	OR		
XII	a) • In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method. • In call by value method, we can not modify the value of the actual parameter by the formal parameter. • In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter. • The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition. • In call by reference, the address of the variable is passed into the function call as the actual parameter. • The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed. • In call by reference, the memory allocation is similar for both formal parameters and actual parameters.		

	• All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address. ANY 3 differences 3marks b) Create a function to check prime number with exact logic-2 marks Calling function and displaying result-1mark		

XIII	#include <stdio.h>
	void main()
	{
	
	int i, j, a, n, number[30];
	printf("Enter the value of N \n");
	scanf("%d", &n);
	
	printf("Enter the numbers \n");
	for (i = 0; i < n; ++i)
	scanf("%d", &number[i]);
	
	for (i = 0; i < n; ++i)
	{
	
	for (j = i + 1; j < n; ++j)
	{
	
	if (number[i] > number[j])
	{
	
	a = number[i];
	number[i] = number[j];
	number[j] = a;
	
	}
	
	}
	
	}
	
	printf("The numbers arranged in ascending order are given below \n");
	for (i = 0; i < n; ++i)
	printf("%d\n", number[i]);
	
	}
	Exact program-7marks		

	OR		
XIV	#include <stdio.h> void main () {		

```
 int array[10][10];
int i, j, m, n, a = 0, sum = 0;

printf("Enetr the order of the matix \n");
scanf("%d %d", &m, &n);

if (m == n )
{

   printf("Enter the co-efficients of the matrix\n");
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
         scanf("%d", &array[i][j]);
      }
   }

   printf("The given matrix is \n");
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
         printf(" %d", array[i][j]);
      }
      printf("\n");
   }

   for (i = 0; i < m; ++i)
   {
      sum = sum + array[i][i];
   }

   printf("\nThe sum of the main diagonal elements is = %d\n", sum);
}

else
   printf("The given order is not square matrix\n");

}
```

Exact program-7 marks

```
 int array[10][10];
int i, j, m, n, a = 0, sum = 0;

printf("Enetr the order of the matix \n");
scanf("%d %d", &m, &n);

if (m == n )
{

   printf("Enter the co-efficients of the matrix\n");
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
         scanf("%d", &array[i][j]);
      }
   }

   printf("The given matrix is \n");
   for (i = 0; i < m; ++i)
   {
      for (j = 0; j < n; ++j)
      {
         printf(" %d", array[i][j]);
      }
      printf("\n");
   }

   for (i = 0; i < m; ++i)
   {
      sum = sum + array[i][i];
   }

   printf("\nThe sum of the main diagonal elements is = %d\n", sum);
}

else
   printf("The given order is not square matrix\n");

}
```

Exact program-7 marks

Blue Print

Mark Distribution

Module	Hours/Module (hi)	Marks/Module ($h^i/\sum H^i$) * 123 (±5%)	Type of Questions							
			Part A		Part B		Part C		Total	
			No. of questions	Marks	No. of questions	Marks	No. of questions	Marks	No. of questions	Marks
1	11	31	2	2	2	6	2	14	6	22
2	11	31	3	3	3	9	4	28	10	40
3	10	30	2	2	3	9	4	28	9	39
4	11	31	2	2	2	6	2	14	6	22
Total	43	123	9	9	10	30	12	84	31	123

Cognitive Level Distribution

Cognitive Level	Marks	% of Marks
Remembering	50	41
Understanding	38	31
Applying	35	28
Total	123	100

BLUE PRINT

Mark Distribution

Module	Hr / Module	(hi / ∑Hi) * 123	TYPE OF QUESTIONS							
			PART A		PART B		PART C		TOTAL	
			No of Questions	Marks	No of Questions	Marks	No of Questions	Marks	No of Questions	Marks
I	8	22.8	3	3	3	9	4	28	10	40
II	15	43.05	2	2	3	9	4	28	10	39
III	8	22.8	2	2	2	6	2	14	6	22
IV	12	33.21	2	2	2	6	2	14	6	22
Total	43	123	9	9	10	30	12	84	32	123

Cognitive Level Wise Question Analysis

Mark Distribution

Cognitive Level	% Marks	Marks	TYPE OF QUESTIONS							
			PART A		PART B		PART C		TOTAL	
			No of Questions	Marks	No of Questions	Marks	No of Questions	Marks	No of Questions	Marks
R			3	3	5	15	1	7	9	25
U			4	4	3	9	5	35	12	48
A			2	2	2	6	6	42	10	50
Total	**100**		9	9	10	30	12	84	31	123