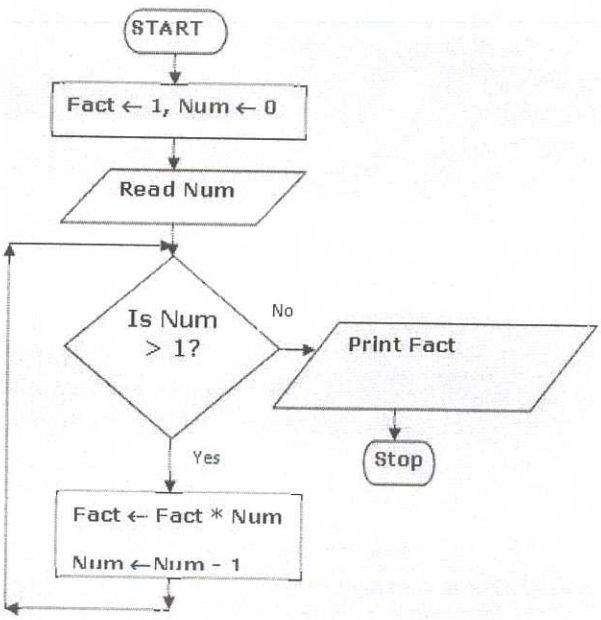
Scoring Indicators

Course Name : Problem Solving and Programming

Course Code : R(21) 2131

QID : 2106220109

Q. No.	Scoring Indicators	Split score	Sub Total	Total score
	PART A			9
I.1	False		1	
I.2	scanf("%d", &num);		1	
I.3	True		1	
I.4	break		1	
I.5	goto		1	
I.6	void		1	
I.7	True		1	
I.8	float num[15];		1	
I.9	An array is a fixed-sized sequenced collection of elements of the same data type that share a common name. An array can be used to represent a list of numbers, or a list of names.		1	
	PART B			24
II.1	Lising – 1 mark Explanations – 2 marks int – Integer char – Character float – Floating Point Numbers double – Double Precision Floating point numbers	1 +2	3	
II.2	Symbol Usage : 1 mark Logic implementation : 2 marks Sample Flowchart  	1 +2	3	

II.3	Syntax : 2 marks Example : 1 mark <u>Syntax</u> if (test condition) { True block statements; ..... } else { False block statements; ..... } <u>Example</u> if (a>b) { printf("The largest number is %d\n",a); } else { printf("The largest number is %d\n",b); }	2 + 1	3	
II.4	Explanation : 2 marks Example : 1 mark During the loop operations, it may be necessary to skip a part of the body of the loop under certain conditions. The **continue** statement causes the loop to be continued with the next iteration after skipping any statement in between. <u>Example</u> .... s=0; for (i=0; i < 10; i++) { printf("Enter a positive integer :"); scanf("%d", &n); if(n<0) continue; s=s+n; }	2 + 1	3	

II.5	Variable declaration and data input : 1 mark Logic and output : 2 marks Sample Program #include <stdio.h> int main() { int first,second; printf("Enter the first number : "); scanf("%d",&first); printf("Enter the second number : "); scanf("%d",&second); if (first % second == 0) printf(" %d is perfectly divisible by %d\n ",first, second); else printf("%d is not perfectly divisible by %d\n",first,second); return 0; }	1 + 2	3	
II.6	Variable declaration and initialization : 1 mark Logic and output : 2 marks Sample Program #include <stdio.h> int main() { int i,sum=0; for (i = 1 ; i<=100; i++) { if(i % 2 == 0) sum += i; } printf ("Sum = %d \n",sum); return 0; }	1 + 2	3	
II.7	Explanation : 2 marks Example : 1 mark Actual parameters: The parameters that appear in function calls. Formal parameters: The parameters that appear in function declarations.	2 + 1	3	
II.8	Function declaration :1 mark Function call : 1 mark Function definition : 1 mark Sample Program #include <stdio.h>	1+1+1	3	

	float area (float, float); int main() { float l,b,ar; printf("Enter the length and breadth of the rectangle : "); scanf("%f%f",&l,&b); ar=area(l,b); printf("Area of the rectangle = %f\n",ar); return 0; } int area (float a, float b) { float ar; ar = a * b; return (ar); }			
II.9	Array and variable declarations : 1 mark Reading data : 1 mark display array: 1 mark <u>Sample Program</u> #include <stdio.h> int main() { int n,arr[100],i; printf("Enter the number of elements in the array : "); scanf("%d",&n); printf("Enter the elements: "); for(i=0;i<n;i++) scanf("%d", &arr[i]); printf("The entered array elements are : \n"); for(i=0; i<n; i++) printf("%d\n",arr[i]; return 0; }	1+1+1	3	
II.10	Array declaration : 1 mark Reading data : 2 marks	1 + 2	3	
	PART C			42
III.1(a)	Pre-increment explanation – 1 mark Pre-increment example – 1 mark Post-increment explanation – 1 mark Post-increment example – 1 mark While ++m amd m++ (or --m and m--) mean the same thing when they form statements independently, they behave defferently when	1+2+1	7	

	they are used in expressions on the right-hand side of an assignment statement. Consider the following example: m = 5 y = ++m; In this case, the value of y and m would be 6. Suppose, if we rewrite the above statements as m = 5; y = m++; then, the value of y would be 5 and m would be 6. A prefix operator first adds 1 to the operand and then the result is assigneed to the variable on left. On the otherhand, a postfix operator first assigns the value to the variable on left and then increments the operand.			
III.1(b)	Variable declareation and reading : 1 mark Calculation and output : 2 marks <u>Sample Program</u> #include <stdio.h> int main() { float f,c; printf("Enter the temperature in Fahrenheit : "); scanf("%f",&f); c = (f – 32) * 5 / 9.0; printf("Temperature in Celsius = %f\n",c); return 0; }	1 + 2		
III.2(a)	Variable declareation and reading : 1 mark Calculation and output : 3 marks <u>Sample Program</u> #include <stdio.h> int main() { int minutes, hr, mn; printf("Enter the time in minutes : "); scanf("%d",&minutes); hr=minutes / 3600; mn=minutes % 3600; m=mn/60; s=mn%60; printf("Hours = %d , Minutes = %d secconds = %d ", hr,m,s); return 0; }	4	7	

III.2(b)	1. First character must be an alphabet (or underscore). 2. Must consist of only letters, digits or underscore. 3. Only first 31 characters are significant. 4. Cannot use a keyword. 5. Must not contain white space.	3		
III.3	Explanation : 2 marks Syntax : 3 marks Example : 2 marks **switch** statement is a multiway decision statement in C language. The **switch** statement tests the value of a given variable (or expression) against a list of **case** values and when a match is found, a block of statements associated with that **case** is executed. The expression is an integer expression or characters. Value-1, Value-2, ... are constants. The break statement at the end of each block signals the end of a particular case and causes an exit from the **switch** statement. Syntax switch(Expression) { case value-1: block-1 statements; break; case value-2: block-2 statements; break; case value-3: block-3 statements; break; .... default: default-block statementts; Example .... switch(n) { case 0: printf("Zero\n"); break; case 1: printf("One\n"); break; case 2: printf("Two\n"); break;	2+3+2	7	

	.... }			
III.4	Variable declaration and reading : 2 Logic : 4 marks Output : 1 mark <u>Sample Program</u> #include <stdio.h> int main() { int num,num1,d,sum=0; printf("Enter a number : "); scanf("%d",&num); num1=num; while(num1 > 0) { d = num1 % 10; sum += d*d*d; num1 /= 10; } if(num = = sum) printf("The given number is armstrong\n"); else printf("The given number is not a armstrong no\n"); return 0; }	2+4+1	7	
III.5	Explanations :1 marks Syntax : 1 mark each (3) Examples : 1 mark each (3) <u>while loop</u> The simplest of all the looping structures in C is the **while** statement. The basic format of the while statement is; while (test – condition) { body of the loop } The while is an entry-controlled loop statement. Eg:	1+3+3	7	

```
while (n <=10)
        {
                sum = sum + n;
                n= n + 1;
        }
.....
.....
```

<u>do...while loop</u>

In some occasions it might be necessary to execute the body of the loop before the test is performed. Such situations can be handled with the help of the do ... while statement.

```
do
{
        body of the loop

} while (test-condition);
```

Eg:

```
.....
.....
do
        {
                sum = sum + n;
                n= n + 1;
        }
        while (n<=10);
.....
.....
```

<u>for loop</u>

The for loop is another entry-controlled loop that provides a more concise loop control structure. The general form of the **for** loop is;

```
for ( initialization  ;  test-condition   ;  increment/decrement )
{
        body of the loop
}
```

Eg:

```
.....
.....
for (n=1 ; n <= 10 ; n = n +1)
        {
                sum = sum + n;
        }
.....
.....
```

III.6	Variable declaration and reading : 2 Logic : 4 marks Output : 1 mark Sample Program #include <stdio.h> int main() { float unit,charge; printf("Enter the unit consumed : "); scanf("%f",&unit); if(unit <=150) charge = unit * 3; else if (unit <=350) charge = 450 + (unit-150) * 3.75; else if (unit <=450) charge = 1200 + (unit-350) * 4; else charge = 1600 + (unit-450) * 5; printf("Unit Consumed : %f \n",unit); printf(Chage :%f\n",charge); return 0; }	2+4+1	7	
III.7	Function declaration : 2 marks Function call and output : 2 marks Function definition : 3 marks Sample Program #include <stdio.h> #include <math.h> float cal_area(int,int,int); int main() { int a,b,c; float area; printf("Enter the three sides of a triangle : "); scanf("%d%d%d",&a,&b,&c); area = cal_area(a,b,c); printf("Area of the triangle: %f \n",area); return 0; } float cal_area(int x, int y, int z) { float a,s;	2+2+3	7	

	s=(x+y+z) / 2.0; a= sqrt (s * (s-x) * (s-y) * (s-z)) return a; }			
III.8	Explanations : 5 marks Example : 2 marks In order to make use of a user-defined function, we need to establish three elements that are related to functions. 1. Function definition 2. Function call 3. Function declaration <u>Function Definition</u> The function definition is an independent program module that is specially written to implement the requirements of the function. A function definition, also known as function implementation shall include the following elements. 1. Function return type 2. Function name 3. List of parameters 4. local variable declarations 5. function statements 6. a return statement All the six elements are grouped into two parts, namely, - function header (first three elements) - function body (second three elements) <u>Function Call</u> A function can be called by simply using the function name followed by a list of actual parameters (or arguments), if any, enclosed in parentheses. <u>Function Declaration</u> All functions in a C program must be declared, before they are invoked. A function declaration (also known as function prototype) consists of four parts. 1. Function type (return type) 2. Function name 2. Parameter list 4. Terminating semicolon <u>Example program:</u> #include <stdio.h> int mul(int, int); // Function Declaration (Function prototype) int main()	5 + 2	7	

	{ int a,b,result; printf("Enter two nos : "); scanf("%d%d",&a,&b); result=mul(a,b); // Function call printf("Product = %d\n",result); return(0); } int mul (int x, int y) // Function Definition { int q; q=x * y; return(q); }			
III.9	Array and variable declarations : 1 mark Reading data : 1 mark sorting logic and printing: 5 marks <u>Sample Program</u> #include <stdio.h> int main() { int n,arr[100],i,j,t; printf("Enter the number of elements in the array : "); scanf("%d",&n); printf("Enter the elements: "); for(i=0;i<n;i++) scanf("%d", &arr[i]); for(i=0;i<n;i++) for(j=i+1;j<n;j++) if(arr[i] > arr[j]) { t = arr[i]; arr[i] = arr[j]; arr[j] = t; } printf("The sorted array elements are : \n"); for(i=0; i<n; i++) printf("%d\n",arr[i]; return 0; }	1+1+5	7	
III.10	Array and variable declarations : 1 mark Reading data : 1 mark program logic and printing: 5 marks	1+1+5	7	

	Sample Program #include <stdio.h> int main() { int n,arr[100],i, even=0,odd=0; printf("Enter the number of elements in the array : "); scanf("%d",&n); printf("Enter the elements: "); for(i=0;i<n;i++) scanf("%d", &arr[i]); for(i=0;i<n;i++) { if (arr[i] %2 == 0) even++; else odd++; } printf("Count of even numbers : %d \n",even); printf("Count of odd numbers : %d \n",odd); return 0; }			
III.11	Array and variable declarations : 2 marks Data input : 1 mark Logic and display : 4 marks Sample Program #include <stdio.h> int main() { int a[10][10], r,c,i,j,l; printf("Enter the order of matrices : "); scanf("%d%d",&r,&c); printf("Enter the elements of matrix :\n"); for(i=0;i<r,i++) for(j=0;j<c;j++) scanf("%d",&a[i][j]); l=a[0][0]; for(i=0;i<r,i++) for(j=0;j<c;j++) if (l < a[i] [j]) l=a [i] [j]; printf("The largest number in the matrix : %d \n",l); return 0; }	2+1+4	7	
III.12	Array and variable declarations : 2 marks	2+1+4	7	

Data input : 1 mark
Logic and display : 4 marks

Sample Program

```
#include <stdio.h>
int main()
{
        int a[10][10], r,c,i,j,s = 0;

        printf("Enter the order of matrices : ");
        scanf("%d%d",&r,&c);
        if(r !=c)
                printf("The order is not a square matrix\n");
        else
        {
                printf("Enter the elements of matrix :\n");
                for(i=0;i<r,i++)
                        for(j=0;j<c;j++)
                                scanf("%d",&a[i][j]);

                for(i=0;i<r,i++)
                {
                        s += a[i][i];
                }
                printf("The sum of main diagonal = %d\n",s);
        }
        return 0;
}
```