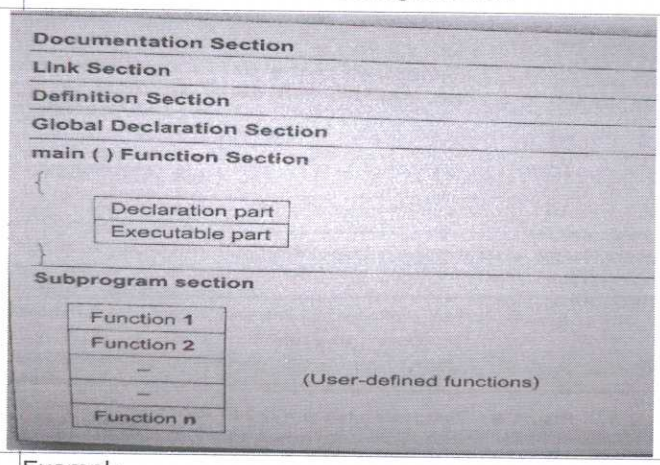
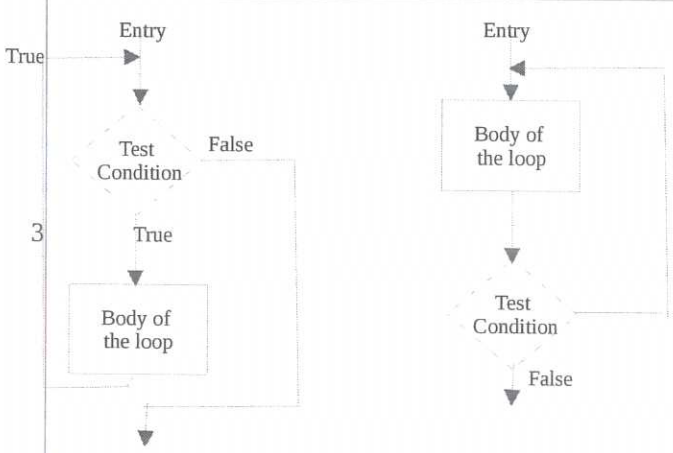
Scheme of Valuation					
Scoring Indicator					
Revision: 2015				**Course Code: 4044**	
Course Titile: Programming in C					
Qst No.		**Scoring Indicator**	**Splitup Score**	**Sub Total**	**Total**
I	1	(condition) ? expression1 : expression2	1	2	10
		Example	1		
	2	0 2 4 6 8	2		
	3	strcmp(string 1, string2); - compares its first and second arguments character wise. It returns 0 if the strings are equal. It returns a positive value if the It returns a negative value if the first string is less than second. First string is greater than second.	2		
	4	char *str = "Hello";	2		
	5	exit is used terminate the program. return is a statement that returns control back to the calling function.	2		
II			3	6	
		Example	3		

	2	switch(expression) {
		case constant-expression :
		statement(s);
		break; /* optional */
		
		case constant-expression :
		statement(s);
		break; /* optional */
		
		/* you can have any number of case statements */
		default : /* Optional */
		statement(s);
		} description
			4	6	

		Example	2		

3	 **Entry Controlled Loop:** Test condition is checked first, and then loop body will be executed. If Test condition is false, loop body will not be executed. for loop and while loop are the examples of Entry Controlled Loop. Entry Controlled Loops are used when checking of test condition is mandatory before executing loop body. **Exit Controlled Loop:** Loop body will be executed first, and then condition is checked. If Test condition is false, loop body will be executed once. do while loop is the example of Exit controlled loop. Exit Controlled Loop is used when checking of test condition is mandatory after executing the loop body.		6

4	char s[1000];
	int i;
	printf("Enter a string: ");
	scanf("%s", s);
	for(i = 0; s[i] != '\0'; ++i);
	printf("Length of string: %d", i);
	Or similar program
		6	6

5	size_t strlen(const char *str) : It returns the length of the string without including end character int strcmp(const char *str1, const char *str2) : It compares the two strings and returns an integer value char *strcat(char *str1, char *str2) It concatenates two strings and returns the concatenated string. char *strcpy(char *str1, char *str2) It copies the string str2 into string str1, including the end character or any other string fuction with syntax decription	Description 3marks Syntax 3marks	6

6	return_type function_name (argument list)
	{
	Set of statements – Block of code
	}
	return_type: Return type can be of any data type such as int, double, char, void, short etc. Don't worry you will understand these terms better once you go through the examples below.
	function_name: It can be anything, however it is advised to have a meaningful name for the functions so that it would be easy to understand the purpose of function just by seeing it's name.
		2	6

30

		argument list: Argument list contains variables names along with their data types. These arguments are kind of inputs for the function. For example – A function which is used to add two integer variables, will be having two integer argument. Block of code: Set of C statements, which will be executed whenever a call will be made to the function. return statement: The return statement terminates the execution of a function and returns a value to the calling function. The program control is transferred to the calling function after return statement.	4		
	7	The variables declared in the function prototype or definition are known as **Formal arguments** and the values that are passed to the called function from the main function are known as **Actual arguments**. Description with example	2 4	6	

III	a	if (num1 > num2)
		{
		if (num1 > num3)
		{
		printf("num1 is the greatest among three \n");
		}
		else
		{
		printf("num3 is the greatest among three \n");
		}
		}
		else if (num2 > num3)
		printf("num2 is the greatest among three \n");
		else
		printf("num3 is the greatest among three \n");
		Or similar program
			9		15

	b	When an if else statement is present inside the body of another "if" or "else" then this is called nested if else.
		Syntax of Nested if else statement:
		if(condition) {
		//Nested if else inside the body of "if"
		if(condition2) {
		//Statements inside the body of nested "if"
		}
		else {
		//Statements inside the body of nested "else"
		}
		}
		else {
		//Statements inside the body of "else"
		}
		Example
			1
			3
			2
				6	

IV	a	Arithmetic Operators Relational Operators Logical Operators Assignment Operators Increment and Decrement Operators Conditional Operator Bitwise Operators	3	9	

		Special Operators Description of each	6		

	b	`float radius, height,PIE=3.14;`
		`float volume;`
		`scanf("%f", &radius);`
		`printf("Enter value for height of a cylinder : \n");`
		`scanf("%f", &height);`
		`volume = PIE * radius * radius * height;`
		`printf("\nVolume of cylinder is : %f", volume);`
			6		15

V	a	`scanf("%d",&number);`
		`printf("\n%d %d",n1,n2);`
		`for(i=2;i<number;++i)`
		`{`
		`n3=n1+n2;`
		`printf(" %d",n3);`
		`n1=n2;`
		`n2=n3;`
		`}`
			8		

	b	The simplest form of multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows – type arrayName [x][y]; array initialization accessing elements in two dimensional array	2		15
			3	7	
		example	2		

VI	a	for loop : syntax
		for (initialization; condition test; increment or decrement)
		{
		//Statements to be executed repeatedly
		}
		Description about initalization , condition test and increment or decrement
		example
			3		

		While loop : syntax
		while(condition) {
		statement(s);
		}			

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

		statement(s) may be a single statement or a block of statements. The conditionmay be any expression, and true is any nonzero value. The loop iterates, while the condition is true. When the condition becomes false, the program control passes to the line immediately after the loop. example	3	9	15

		do...while loop : syntax
		do {
		statement(s);
		} while(condition);
		the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested. If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again.
			3		

	b	`int arr[10], i, n, sum = 0;`
		`printf("Enter n: ");`
		`scanf("%d", &n);`
		`for(i=0; i<n; ++i)`
		`{`
		`scanf("%d", &arr[i]);`
		`sum += arr[i];`
		`}`
		`printf("Sum is = %d", sum);`
			6		

VII	a	`char s1[100], s2[100], i, j,len1,len2`
		`printf("Enter first string: ");`
		`scanf("%s", s1);`
		`printf("Enter second string: ");`
		`scanf("%s", s2);`
		`len1=strlen(s1);`
		`len2=strlen(s2);`
		`for(i=len-1,j = 0; j<=len2;++j, ++i)`
		`{`
		`s1[i] = s2[j];`
		`}`
		`printf("After concatenation:%s",s1);`
			9		15

	b	Pointers are used in C program to access the memory and manipulate the address. Address concept in C **pointer variable decleration** data_type* pointer_variable_name; int* p; & and * operator example	4 2	6	
VIII	a	a string is an array of characters terminated with a null character \0 with example Initialization methods with example char c[] = "abcd"; char c[50] = "abcd";	2 4	6	

		char c[] = {'a', 'b', 'c', 'd', '\0'}; char c[5] = {'a', 'b', 'c', 'd', '\0'};			15

	b	s=&a[0];
		small=*s;
		for(i=0;i<5;i++,s++)
		if(*s<small)
		small=*s;
		printf("\nSmallest Element :%d",small);
			9		

IX	a	1. Function with no arguments and no return value with eg: 2. Function with no arguments and a return value with eg: 3. Function with arguments and no return value with eg: 4. Function with arguments and a return value with eg:	2 2 2 2	8	

	b	scanf("%d",&n);
		f= fact(n);
		printf("\n Factorial of %d is %d",n, fact);
		int fact(int n)
		{
		if(n==1)
		Return 1;
		return n*fact(n-1);
		}
			7		15

X	a	call by value: In call by value, a copy of actual arguments is passed to formal arguments of the called function and any change made to the formal arguments in the called function have no effect on the values of actual arguments in the calling function. + example	2+2	8	
		call by reference: In call by reference, the location (address) of actual arguments is passed to formal arguments of the called function. This means by accessing the addresses of actual arguments we can alter them within from the called function.+ example	2+2		

	b	int addNumbers(int a, int b);
		int main()
		{
		int n1,n2,sum;
		printf("Enters two numbers: ");
		scanf("%d %d",&n1,&n2);
		sum = addNumbers(n1, n2);
		printf("sum = %d",sum);
		return 0;
		}
		int addNumbers(int a,int b)
		{
		int result;
		result = a+b;
		return result;
		}
			7		15