SCHEME OF VALUATION

(Scoring Indicators)

Revision: 2015 Course Code: 2131

Course Title: PROGRAMMING IN C

Qst. No.	Scoring Indicator	Split up score	Sub Total	Total
	PART A			
I 1	. A **variable** in simple terms is a storage place which has some memory allocated to it. Basically, a variable used to store some form of data. Different types of variables require different amounts of memory, and have some specific set of operations which can be applied on them.	2	2	10
2	A function prototype is a declaration in C and C++ of a function, its name, parameters and return type before its actual declaration. This enables the compiler to perform more robust type checking. Because the function prototype tells the compiler what to expect, the compiler is better able to flag any functions that don't contain the expected information. A function prototype omits the function body.	2	2	
3	An array of arrays is known as 2D array. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns. Before we discuss more about two Dimensional array lets have a look at the following C program.	2	2	
4	A structure is a user-defined data type available in C that allows to combining data items of different kinds. Structures are used to represent a record. **Defining a structure:** To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than one member.	2	2	
5	C conditional statements allow you to make a decision, based upon the result of a condition. These statements are called Decision Making Statements or Conditional Statements.	0.5x4	2	
II . 1	1. **Entry Controlled loops**: In this type of loops the test condition is tested before entering the loop body. **For Loop**and 2. **While Loop** are entry controlled loops.	3 3	6	30

2	Switch case statements are a substitute for long if statements that compare a variable to several integral values
	• The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.
	• Switch is a control statement that allows a value to change control of execution.
	Syntax:
	switch (n)
	{
	case 1: // code to be executed if n = 1;
	break;
	case 2: // code to be executed if n = 2;
	break;
	default: // code to be executed if n doesn't match any cases
	}
		3
		3
			6	

3	A function is a set of statements that take inputs, do some specific computation and produces output.
	The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function.
	Example:
	Below is a simple C program to demonstrate functions in C.
	#include <stdio.h>
	// An example function that takes two parameters 'x' and 'y'
	// as input and returns max of two input numbers
	int max(int x, int y)
	{
	if (x > y)
	return x;
	else
	return y;
	}
		3
		3
			6	

4.	To access address of a variable to a pointer, we use the unary operator **&** (ampersand) that returns the address of that variable. For example &x gives us address of variable x. One more operator is **unary** * (Asterisk) which is used for two things : • To declare a pointer variable: When a pointer variable is declared in C/C++, there must a * before its name. To access the value stored in the address we use the unary operator (*) that returns the value of the variable located at the address	3 3	6	

	specified by its operand.			

5.	**Array declaration in C/C++:**
	We can declare an array by specifying its type and size or by initializing it or by both.
	1. **Array declaration by specifying size**
	// Array declaration by specifying size
	int arr1[10];
	
	// With recent C/C++ versions, we can also
	// declare an array of user specified size
	int n = 10;
	int arr2[n];
	2. **Array declaration by initializing elements**
	// Array declaration by initializing elements
	int arr[] = { 10, 20, 30, 40 }
		3
		
		3
			6	30

6	Declaration Input Output	2 2 2	6	
7.	Syntax: data_type (*var_name)[size_of_array]; Example: int (*ptr)[10]; Here *ptr* is pointer that can point to an array of 10 integers. Since subscript have higher precedence than indirection, it is necessary to enclose the indirection operator and pointer name inside parentheses. Here the type of ptr is 'pointer to an array of 10 integers'.	3 3	6	

III	(a)	**Arithmetic Operators**: These are the operators used to perform arithmetic/mathematical operations on operands. Examples: (+, -, *, /, %,++,–). Arithmetic operator are of two types: 1. **Unary Operators**: Operators that operates or works with a single operand are unary operators. For example: (++ , –) 2. **Binary Operators**: Operators that operates or works with two operands are binary operators.For example: (+ , – , * , /) • **Logical Operators**: Logical Operators are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a boolean value either true or false. To learn about different logical operators in details please visit this link.	4 4	8	15
	b)	Declaration Inupt Loop output	1 1 3 2	7	
IV	(a)	Explain about exit controlled loops with syntax If Test condition is false, **loop** body will be executed once. **for loop** and while **loop**are the **examples** of **Entry Controlled Loop**. do while **loop** is the example of **Exit controlled loop**. **Entry Controlled Loops** are used when checking of test condition is mandatory before executing **loop** body. Syntax+Expl 4+4	4+4	8	15
	b)	Declaration Inupt Loop output	1 1 3 2	7	
V	(a)	Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the	4	8	15

		function.
		`void recursion() {`
		`  recursion(); /* function calls itself */`
		`}`
		`int main() {`
		`  recursion();`
		`}`
			4		

	b	Call by value method of passing Call by reference method of passing	3 4	7	
VI	(a)	Call by value in C o 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. o In call by value method, we can not modify the value of the actual parameter by the formal parameter. o 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. o 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. Call by reference in C o In call by reference, the address of the variable is passed into the function call as the actual parameter. o 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	4 4	8	15

		parameters, and the modified value gets stored at the same address			
	(b)	**There are four storage classes in C they are as follows:** Automatic **Storage Class.** Register **Storage Class.** Static **Storage Class.** External **Storage Class.** 1. **auto**: This is the default storage class for all the variables declared inside a function or a block. Hence, the keyword auto is rarely used while writing programs in C language. Auto variables can be only accessed within the block/function they have been declared and not outside them (which defines their scope). Of course, these can be accessed within nested blocks within the parent block/function in which the auto variable was declared. However, they can be accessed outside their scope as well using the concept of pointers given here by pointing to the very exact memory location where the variables resides. They are assigned a garbage value by default whenever they are declared. 1. **extern**: Extern storage class simply tells us that the variable is defined elsewhere and not within the same block where it is used. Basically, the value is assigned to it in a different block and this can be overwritten/changed in a different block as well. So an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. It can be accessed within any function/block. Also, a normal global variable can be made extern as well by placing the 'extern' keyword before its declaration/definition in any function/block. This basically signifies that we are not initializing a new variable but instead we are using/accessing the global variable only. The main purpose of using extern variables is that they can be accessed between two different files which are part of a large program. For more information on how extern variables work, have a look at this link. 1. **static**: This storage class is used to declare static variables which are popularly used while writing programs in C language. Static variables have a property of preserving their value even after they are out of their scope! Hence, static variables preserve the value of their last use in their scope. So we can say that they are initialized only once and exist till the termination of the program. Thus, no new memory is allocated because they are not re-declared. Their scope is local to the function to which they were defined. Global static variables can be accessed anywhere in the program. By default, they are assigned the value 0 by the compiler. 1. **register**: This storage class declares register variables which have the same functionality as that of the auto variables. The only difference is that the compiler tries to store these variables in the register of the microprocessor if a free register is available. This makes the use of register variables to be much faster than that of the variables stored in the memory during the runtime of the program. If a free register is not available,	4+3	7	

		these are then stored in the memory only. Usually few variables which are to be accessed very frequently in a program are declared with the register keyword which improves the running time of the program. An important and interesting point to be noted here is that we cannot obtain the address of a register variable using pointers. Any two			
VII	(a)	Declaration Input Loop output	1 1 4 2	8	15

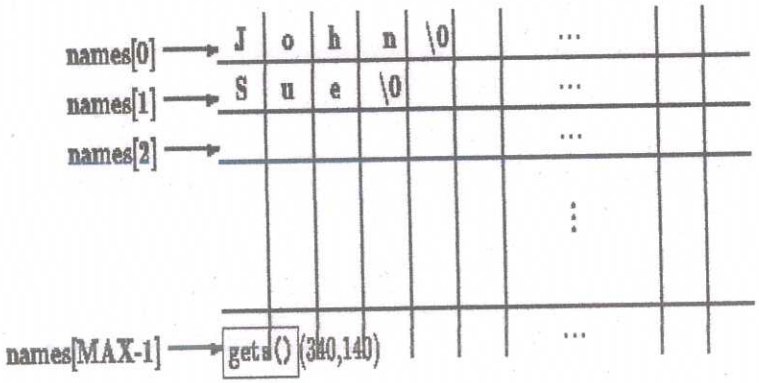
	(b)	**Passing an entire array to a function**
		// Program to calculate average by passing an array to a function
		
		#include <stdio.h>
		float average(float age[]);
		
		int main()
		{
		float avg, age[] = {23.4, 55, 22.6, 3, 40.5, 18};
		avg = average(age); // Only name of an array is passed as an argument
		printf("Average age = %.2f", avg);
		return 0;
		}
		
		float average(float age[])
		{
		int i;
		float avg, sum = 0.0;
		for (i = 0; i < 6; ++i) {
		sum += age[i];
		}
		avg = (sum / 6);
			1
			3
			3
				7	

VIII	(a)	Declaration Input Loop output	1 1 4 2	8	15

	(b)	**Arrays** and **pointers** are synonymous in terms **of** how they use **to** access memory. But, the important **difference between** them is that, a **pointer** variable can take different addresses as value whereas, in case **of array** it is fixed. **In C** , name **of** the**array** always points **to** the first element **of** an **array**.	4+3	7	

IX	(a)	The **strlen()** function calculates the length of a given string.The **strlen()** function is defined in **string.h** header file. It doesn't count null character '\0'.
		Syntax:
		int strlen(const char *str);
		
		strcpy() is a standard library function in C/C++ and is used to copy one string to another. In C it is present in **string.h** header file and in C++ it is present in **cstring** header file.
		Syntax:
		char* strcpy(char* dest, const char* src);
		
		strcmp() in C/C++
		
		strcmp() is a built-in library function and is declared in **<string.h>** header file. This function takes two strings as arguments and compare these two strings lexicographically.
		Syntax::
		int strcmp(const char *leftStr, const char *rightStr);
		
		In the above prototype, function srtcmp takes two strings as parameters and returns an integer value based on the comparison of strings.
		• strcmp() compares the **two strings lexicographically** means it starts comparison character by character starting from the first character until the characters in both strings are equal or a NULL character is encountered.
		• If first character in both strings are equal, then this function will check the second character, if this is also equal then it will check the third and so on
		• This process will be continued until a character in either string is NULL or the characters are unequal.
			3
			
			3
			
			3
				9	15

	(b)	Besides data base applications, another common application of two dimensional arrays is to store an array of strings. In this section we see how an array of strings can be declared and operations such as reading, printing and sorting can be performed on them. A string is an array of characters; so, an array of strings is an array of arrays of characters. Of course, the maximum size is the same for all the strings stored in a two dimensional array.	2	6	

		We can declare a two dimensional character array of MAX strings of size SIZE as follows: `char names[MAX][SIZE];` Since `names` is an array of character arrays, `names[i]` is the character array, i.e. it points to the character array or string, and may be used as a string of maximum size `SIZE - 1`. As usual with strings, a NULL character must terminate each character string in the array. We can think of an array of strings as a table of strings, where each row of the table is a string as seen in Figure 9.13.  Figure 9.13: An Array of Strings	2 2	6	
X		Declaration Input Loop output	5 2 5 3	15	15

CODE – 2131 (15)

COURSE: PROGRAMMING IN C

VERSION:1

BLUE PRINT

SI No	Module	Type of Questions							
		Part A		Part B		Part C		Total	
		No. of Questions	Score	No. of Questions	Score	No. of Questions	Score	No. of Questions	Score
1	Module 1	2	4	2	12	4	30	8	46
2	Module 2	1	2	2	12	4	30	7	44
3	Module 3	1	2	2	12	4	30	7	44
4	Module 4	1	2	1	6	4	30	6	38
	Total	5	10	7	42	16	120	28	172