SCHEME OF VALUATION				
(Scoring Indicators)				
Revision: 2021				
Course Name: Programming in C				
Course Code: 3132		QID:		
Qst. No	Scoring Indicator	Split up Score	Sub Total	Total
I	PART A			9
1)	?:		1	
2)	void		1	
3)	extern		1	
4)	int a[3][2]={{1,2},{2,3},{3,4}};		1	
5)	strlen(), strcpy(), strcat(), strcmp(), strrev().. Any two		1	
6)	pointer		1	
7)	ptr=(type*)malloc(size); eg: ptr = (int*)malloc(n*sizeof(int));		1	
8)	using dot[.] operator, structure_object.member		1	
9)	fscanf(), fgets(), fread() any one		1	
II	PART B			24
1)	Preprocessors are programs that process our source code before compilation. All of these preprocessor directives begin with a '#' (hash) symbol. Some preprocessor directives are: #include, #define etc. #include is used to include header file which is necessary to use built-in functions in the program. Eg: #include<stdio.h> includes header file stdio.h which contains printf() and scanf() functions		3	
2)	An array is a homogeneous sequential collection of data items over a single variable name.An array is always stored in consecutive memory location.It can store multiple value of similar type, which can be referred with single name.An array can either be an integer, character, or float data type.All elements of an array can be distinguished with the help of index number.		3	

3)	The output of the program is 3, 2, 15 a[0] = 5, a[1] = 1, a[2] = 15, a[3] = 20, a[4] = 25 i = ++a[1]; becomes i = ++1; Hence i = 2 and a[1] = 2 j = a[1]++; becomes j = 2++; Hence j = 2 and a[1] = 3. m = a[i++]; becomes m = a[2]; Hence m = 15 and i is incremented by 1(i++ means 2++ so i=3) printf("%d, %d, %d", i, j, m); It prints the value of the variables i, j, m		3	

4)	strcmp() is used to check whether two strins are equal or not. The function returns zero if the strings are equal.
	If(strcmp(s1,s2)==0)
	{
	Printf("\nStrings are equal\n");
	}
			3	

5)	return_type function(type arrayname[]) Declaring blank subscript notation [] is the widely used technique. array size can also be specified as follows, return_type function(type arrayname[SIZE])		3	

6)	The call by reference method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. It means the changes made to the parameter affect the passed argument.
	void swap(int *x, int *y) {
	int temp;
	temp = *x;
	*x = *y;
	*y = temp;
	return;
	}
			3	

7)	This allocation method has no memory wastage. The memory allocation is done at run time. Memory size can be changed based on the requirements of the dynamic memory allocation. If memory is not required, it can be freed.		3	

2)	#include <stdio.h>
	int addNumbers(int n);
	int main() {
	int num;
	printf("Enter a positive integer: ");
	scanf("%d", &num);
	printf("Sum = %d", addNumbers(num));
	return 0;
	}
	int addNumbers(int n) {
	if (n != 0)
	return n + addNumbers(n - 1);
	else
	return n;
	}
			7	

3)	a) The output is Count=1 Count=2 Count=3 Static variable count retains its value through out the program. b) Static variables are local variables. A static int variable remains in memory while the program is running. Static variables are initialized as 0 When the value is changed it reflects on further use of that variable.	4 3	7	
	OR			

10)	The command line arguments are values passed to the main program during execution. int main(int argc, char *argv[]) The command line arguments are handled using main() function arguments where argc refers to the number of arguments passed, and argv[] is a pointer array which points to each argument passed to the program.		7	

11)	User-defined data types are created by the user using a combination of fundamental and derived data types in the C programming language.
	Structures are used to group multiple variables of different data types.
	struct <name> {
	<data type 1> <variable name 1>,
	<data type 2> <variable name 2>,
	...
	};
	Example
	struct Book {
	// Struct Book Declaration.
	char name[100];
	int number_of_pages;
	char author[100];
	float price;
	};
		2
		4
		1
			7	

OR				

4)	Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program. C language uses 4 storage classes **auto**:It is a default storage class,variables are local variables, A keyword auto is used to define an auto storage class. By default, an auto variable contains a garbage value. **register**: It is a variable which is stored inside a Register. The keyword register is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program. The variable is limited to the particular block **Static** variables are local variables. A static int variable remains in memory while the program is running. Static variables are initialized as 0. When the value is changed it reflects on further use of that variable. **extern**: an external variable is a variable defined outside any function block. keyword extern is used to define external storage class. these are global variables.		7	
5)	Binary Search is a search algorithm that is used to find the position of an element in a sorted array. The binary search algorithm works by comparing the element to be searched by the middle element of the array and based on this comparison follows the required procedure. Case 1 – element = middle, the element is found return the index. Case 2 – element > middle, search for the element in the sub-array starting from middle+1 index to n. Case 3 – element < middle, search for element in the sub-array starting from 0 index to middle -1. Binary search works efficiently on sorted data no matter the size of the data. After every cycle of search, the algorithm divides the size of the array into half hence in the next iteration it will work only in the remaining half of the array. Example Array: 3 5 7 9 12 15 18 20 23 25 key to be searched is 20 first = 0, last =9 middle=(first+last)/2 = 4 check middle and key, 12 and 20 since 20 is greater than 12 take the sub-array starting from middle +1 continue searching until the key is found or interval is zero	4 3	7	
	OR			

8)	int arr[5] = {100, 200, 300, 400, 500}; int *ptr; ptr = &arr[0]; ptr is an integer pointer which holds the address of the first element. i.e &arr[0] ptr + 1 points the address of second variable. i.e &arr[1] Similarly, ptr + 2 holds the address of the third element of the array. i.e. &arr[2] *ptr is 100, *(ptr+1) is 200 and so on		3	
9)	char strname[array_size]; char str[7]="String"; char str[] = "String"; char str[7]={'S', 't', 'r', 'i', 'n', 'g','\0'}; Explain		3	
10)	Creation of a new file (fopen with attributes as "a" or "a+" or "w" or "w+") Opening an existing file (fopen) Reading from file (fscanf or fgets) Writing to a file (fprintf or fputs) Moving to a specific location in a file (fseek) Closing a file (fclose)		3	
III	**PART C**			42
1)	a) Execution starts at main(). Read a and b. function area is called in the printf statement. the actual parameters a and b is copied into l and b. the function returns l*b. the returned value is printed on screen. sample output Enter length and breadth 4 6 Area=24 b) Parameter passing technique of the above program is call by value. The parameters to be passed is called actual arguments. the copy of actual parameters are copied to formal parameters of function. the formal parameters are local variables of function. the change in those will not reflect on actual parameters	4 3	7	
OR				

8)	Pointers can be used to access arrays. The address of first element of an array can be stored in a pointer to access array. The elements of array can be accessed by simply incrementing pointer. The pointers to array can be passed as an argument to function also		7	

9)	#include <stdio.h>
	struct employee{
	char empname[30];
	char desg[20];
	int empid;
	int basicpay, da, pf,salary;
	} emp[5];
	
	int main()
	{
	int i;
	printf("Enter 5 Employee Details \n \n");
	for(i=0; i<5; i++){
	printf("\nName: ");
	scanf("%s",emp[i].empname);
	printf("\nId: ");
	scanf("%d",&emp[i].empid);
	printf("\Designation: ");
	scanf("%s",emp[i].desg);
	printf("\nBasic pay: ");
	scanf("%d",&emp[i].basicpay);
	emp[i].da= emp[i].basicpay*0.25;
	emp[i].pf= emp[i].basicpay*0.1;
	emp[i].salary= emp[i].basicpay+emp[i].da-emp[i].pf;
	}
	printf("---------- All Employees Details ----------\n");
	for(i=0; i<n; i++){
	printf("Employee Name \t: %s\n",emp[i].empname);
	printf("Employee Id \t: %d\n",emp[i].empid);
	printf("Designation \t: %s\n",emp[i].desg);
	printf("Basic Pay \t: %d\n",emp[i].basicpay);
	printf("DA \t: %d\n",emp[i].da);
	printf("PF\t: %d\n",emp[i].pf);
	printf("Net Salary \t: %d\n",emp[i].salary);
	}
	return 0;
	}
			7	

	OR			

6)	To read a string gets() is used
	char s1[20]; gets(s1);
	Explanation
	strlen() is used to find length of a string.
	it returns integer value.
	l=strlen(s1);
	Explain with example
	strcpy() is used to copy string.
	strcpy(s2,s1);
	Explain with example
	strrev() is used to reverse a string.
	strrev(s2);
	Explain with example
		1
		
		2
		
		2
		
		2
			7	

7)	#include <stdio.h>
	int main()
	{
	int arr[10], *parr;
	int i, j, temp;
	printf("\nPlease Enter 10 elements of Array ");
	for (int i = 0; i < 10; i++)
	{
	scanf("%d", &arr[i]);
	}
	parr = &arr[0];
	for (i = 0; i < 10; i++)
	{
	for (j = i + 1; j < 10; j++)
	{
	if(*(parr + j) < *(parr + i))
	{
	temp = *(parr + i);
	*(parr + i) = *(parr + j);
	*(parr + j) = temp;
	} }}
	printf("\nSorted Array Elements using Pointer \n ");
	for(i = 0; i < Size; i++)
	{
	printf("%d ", *(parr + i));
	}
	return 0;
	}
			7	

	OR			

12)

```
#include <stdio.h>
struct item
{
  int itemid;
  char itemname[20];
  float price;
};
int main()
{
   struct item *itemPtr, it;
   itemPtr = &it;
   printf("Enter item id: ");
   scanf("%d", &itemPtr->itemid);
   printf("Enter item name: ");
  scanf("%s", &itemPtr->iteminame);
   printf("Enter Price: ");
   scanf("%f", &itemPtr->price);
   printf("Displaying:\n");
   printf("Item Id: %d\n", itemPtr->itemid);
    printf("Item Name: %s\n", itemPtr->itemname);
   printf("Price: %f", itemPtr->price);
   return 0;
```

7