Revision 2015 Course Title PROGRAMMING IN C				Course Code 2131
Q.No	Scoring Indicator	Split up score	Sub Total	Total
	PART A			
1	x>10?a=x:a=20;			2
2	C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation. All preprocessor commands begin with a hash symbol (#).			2
3	Datatype arrayname[rows][columns]			2

4.	Struct employee
	{
	int regno;
	char name[10];
	int bpay;
	};
				2

5	strcpy(),strcmp()	2*1		2
	PART B			

1	do {
	statement(s);
	} while(condition);
	
	while(condition) {
	statement(s);
	}
	Explanation 2*1
		2
		
		2
		
		2
				6

2	void main()
	{
	int a;
	printf ("Enter the no.");
	scanf("%d",&a);
	if(a%5==0)
	{
	printf("No.is Divisible by 5");
	flag=1;
	break;
	}
	if (flag==1)
	printf("divisible by 5");
				6

	printf("cube is %d",a*a)
	else
	printf("No is not Divisible by 5");
	
	}			

3	main()
	{
	int x, y, *a, *b, temp;
	printf("Enter the value of x and y\n");
	scanf("%d%d", &x, &y);
	printf("Before Swapping\nx = %d\ny = %d\n", x, y);
	a = &x;
	b = &y;
	
	temp = *b;
	*b = *a;
	*a = temp;
	
	printf("After Swapping\nx = %d\ny = %d\n", x, y);
	}
				6

4	A function that calls itself is known as a recursive function. The recursion continues until some condition is met to prevent it
	
	int factorial(int i) {
	if(i <= 1) {
	return 1;}
	return i * factorial(i - 1);
	}
	main() {
	int i ;
	scanf("%d",&i);
	printf("Factorial of %d is %d\n", i, factorial(i));
	}
		2
		
		4
				6

5	float average(float age[]);
	main()
	{
	float avg, age[] = {23.4, 55, 22.6, 3, 40.5, 18};
	avg = average(age);
	printf("Average age = %.2f", avg);
	}
	float average(float age[])
	{ int i;
	float avg, sum = 0.0;
	for (i = 0; i < 6; ++i) {
	sum += age[i];
	}
	avg = (sum / 6);
	return avg;}
				6

6	main()
	{
	int n, i;
	float sum = 0, x;
	printf("Enter number of elements: ");
	scanf("%d", &n);
	printf("\n\n\nEnter %d elements\n\n", n);
	for(i = 0; i < n; i++)
	{
	scanf("%f", &x);
	sum += x;
	}
	printf("Average of the entered numbers is = %f", (sum/n));
	return 0;}
				6

7	#include<string.h>
	void main()
	{
	char c[10][20],temp[20];
	int i,j,n;
	printf("\nEnter number of strings: ");
	scanf("%d",&n);
	printf("\nEnter the strings: ");
	for(i=0;i<=n;i++)
	{
	gets(c[i]);
	}
	puts(c[i]);
				6

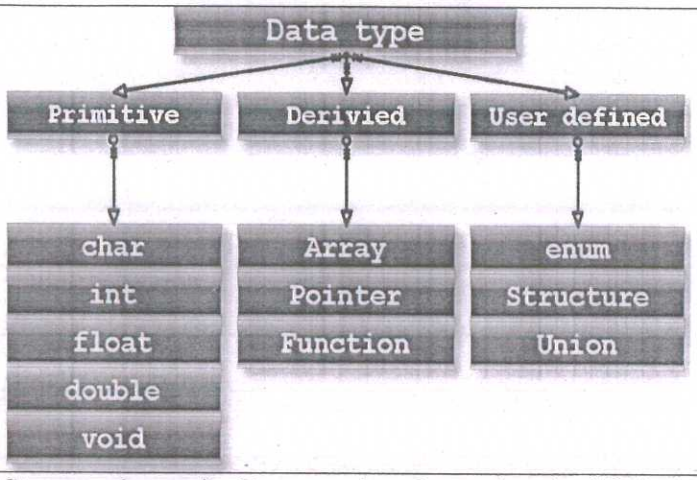
	PART C			

III
a)
	Syntax
	for (init; condition; increment) {
	
	for (init; condition; increment) {
	statement(s);
	}
	statement(s);
	}
		4		8

III (b)	Example.and explanation A variable is an entity whose value keeps on changing throughout the program execution. 1) A Variable name consists of any combination of alphabets, digits and underscores. 2) The first character of the variable name must either be alphabet or underscore. It should not start with the digit. 3) No commas and blanks are allowed in the variable name. 4) No special symbols other than underscore are allowed in the variable name. 5) There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if variable name is longer than 31 characters.	4 1 6		7
IV a)	int main() { int low, high, i, flag;			9

	printf("Enter two numbers(intervals): ");
	scanf("%d %d", &low, &high);
	
	printf("Prime numbers between %d and %d are: ", low, high);
	while (low < high)
	{ flag = 0;
	for(i = 2; i <= low/2; ++i)
	{
	if(low % i == 0)
	{
	flag = 1;
	break;
	}
	}
	if (flag == 0)
	printf("%d ", low);
	++low;
	}
	return 0;}			

IV b)	Ternary Operators takes on 3 Arguments Syntax : expression 1 ? expression 2 : expression 3 where • expression1 is Condition • expression2 is Statement Followed if Condition is True • expression2 is Statement Followed if Condition is False **Example and explanation**	2 4		6

V a)				15
VI a)	Storage classes in C are used to determine the lifetime, visibility, memory location, and initial value of a variable. There are four types of storage classes in C o Automatic o External o Static o Register Automatic o Automatic variables are allocated memory automatically at runtime. o The visibility of the automatic variables is limited to the block in which they are defined. The scope of the automatic variables is limited to the block in which they are defined. o The automatic variables are initialized to garbage by default. o The memory assigned to automatic variables gets freed upon exiting from the block. o The keyword used for defining automatic variables is auto. o Every local variable is automatic in C by default. Static o The variables defined as static specifier can hold	4*2		8

	their value between the multiple function calls. o Static local variables are visible only to the function or the block in which they are defined. o A same static variable can be declared many times but can be assigned at only one time. o Default initial value of the static integral variable is 0 otherwise null. o The visibility of the static global variable is limited to the file in which it has declared. Register o The variables defined as the register is allocated the memory into the CPU registers depending upon the size of the memory remaining in the CPU. o We can not dereference the register variables, i.e., we can not use &operator for the register variable. o The access time of the register variables is faster than the automatic variables. o The initial default value of the register local variables is 0. o The register keyword is used for the variable which should be stored in the CPU register. However, it is compiler?s choice whether or not; the variables can be stored in the register. o We can store pointers into the register, i.e., a register can store the address of a variable. o Static variables can not be stored into the register since we can not use more than one storage specifier for the same variable. o static.\ External o The external storage class is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program. o The variables declared as extern are not allocated			

	any memory. It is only declaration and intended to specify that the variable is declared elsewhere in the program. o The default initial value of external integral type is 0 otherwise null. o We can only initialize the extern variable globally, i.e., we can not initialize the external variable within any block or method. o An external variable can be declared many times but can be initialized at only once. o If a variable is declared as external then the compiler searches for that variable to be initialized somewhere in the program which may be extern or static. If it is not, then the compiler will show an error.			
VI b)	**Call by Value** A copy of actual parameters is passed into formal parameters. Changes in formal parameters will not result in changes in actual parameters. Separate memory location is allocated for actual and formal parameters. **Call by Reference** Reference of actual parameters is passed into formal parameters. Changes in formal parameters will result in changes in actual parameters. Same memory location is allocated for actual and formal parameters. **Example 2*2**	3 4		7

VII a)	void print2largest(int arr[], int arr_size)
	{
	int i, first, second;
	
	/* There should be atleast two elements */
	if (arr_size < 2)
	{
	printf(" Invalid Input ");
	return;
	}
	first = second = INT_MIN;
	for (i = 0; i < arr_size ; i ++)
				15

```
    {
        if (arr[i] > first)
        {
            second = first;
            first = arr[i];
        }

        else if (arr[i] > second && arr[i] != first)
            second = arr[i];
    }
    if (second == INT_MIN)
        printf("There is no second largest element\n");
    else
        printf("The second largest element is %dn", second);
}

int main()
{
    int arr[] = {12, 35, 1, 10, 34, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    print2largest(arr, n);
    return 0;
}
```

VIII a)

```
int main()
{
        int i,j,n,d1=0,d2=0,a[5][5];
        printf("Enter size of square matrix:");
        scanf("%d",&n);
        printf("Enter Elements of matrix:\n");

        for(i=0;i<n;++i)
                for(j=0;j<n;++j)
                {
                        scanf("%d",&a[i][j]);
                        if(i==j)
                                d1+=a[i][j];
                        if((i+j)==(n-1))
                                d2+=a[i][j];
                }

        printf("\nFirst Diagonal Sum=%d",d1);
        printf("\nSecond Diagonal Sum=%d",d2);

        return 0;
}
```

7

VIII b)	Array is collection of homogeneous data	Structure is the collection of heterogeneous data.		
	Array data are access using index.	Structure elements are access using . operator.	4	8
	allocates static memory.	allocate dynamic memory		
	access takes less time than structures.	takes more time than Array.		
	example	**example**	4	

IX

```
struct stu
{
    int rn.,a[5];
    char grade;
    float avg;
 }s[2];

void main()
{
    int i,j,sum,n;
    float avg;

    printf("\t STUDENT MARKSHEET USING
STRUCTURES\n\n");
    printf("Enter the no of students");
    scanf("%d",&n);

for(i=0;i<n;i++)
{
    scanf("%d",s[i].rn);

for(j=0;j<=5;j++)
{
    scanf("%d",&s[i].a[j]);
  }
 }

for(i=0;i<n;i++)
{
    sum=0;

for(j=0;j<5;j++)
{
    sum=sum+s[i].a[j];
    s[i].avg=(float)(sum/5);
    if(s[i].avg>=90)
```

15

	s[i].grade=A;
	else if(s[i].avg>80&&s[i].avg<90)
	s[i].grade=B;
	else
	s[i].grade=C;
	}
	}
	
	printf("**********\n");
	printf("rn\ts1\ts2\ts3\ts4\ts5\tavg\t grade\n");
	printf("**********\n");
	for(i=0;i<n;i++)
	{
	printf("%d\t",s[i].rn);
	for(j=0;j<5;j++)
	{
	printf("%d\t",s[i].a[j]);
	}
	printf("%f\t%d\n",s[i].avg,s[i].grade);
	}
	
	}			

X a)	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2. **strcat(s1, s2);** Concatenates string s2 onto the end of string s1. **strcpy(s1, s2);** Copies string s2 into string s1. **strrev** reverses a given string. Its basic form is given below:	2 2 2 2		8

	strrev(str);			

X b)	Structure is a collection of variables (can be of different types) under a single name.
	
	Keyword struct is used for creating a structure
	
	Syntax of structure
	struct structure_name
	{
	data_type member1;
	data_type member2;
	
	data_type memeber;
	};
	
	Create structure variable
	
	When a structure is defined, it creates a user-defined type. However, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables
	
	struct Person{
	char name[50];
	int citNo;
	float salary;
	} person1, person2, p[20];
	
	Access members of a structure?
	
	Member operator(.)
	
	Eg:
	
	To access salary of person2. person2.salary
		1
		
		2
		
		2
		
		2
				7