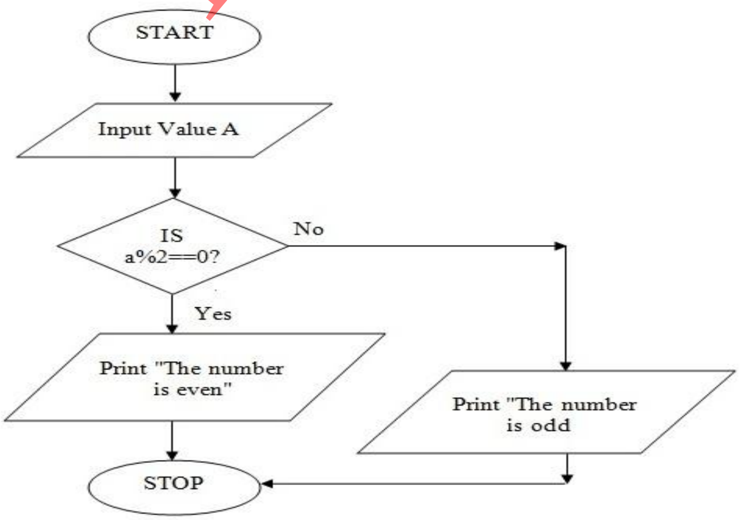
SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Programming in C **Subject Code: 22226**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate’s answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate’s understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	 **(a)** **Ans.**	**Attempt any FIVE of the following:** **Draw flowchart for checking whether given number is even or odd.**  START Input Value A IS a%2==0? No Yes Print "The number is even" Print "The number is odd STOP	**10** **2M** ***Correct logic 1M*** ***Relevant symbol 1M***

(b) **List any four keywords used in 'C' with their use.** **2M**

(Note: Any other relevant keyword in 'C' may be considered).

Ans.

Keyword	Use
auto	It is used to declare auto storage class variable.
break	It is used to exit from block or loop.
case	It is used to represent possible case inside switch case statement
char	Used for declaration of character type variable
const	It is used to declare a constant.
continue	It is used pass control at the beginning of the loop
default	It is used to represent default case inside switch case statement.
do	It is used to execute loop in association with while condition.
double	Used for declaration of double type variable
else	It is used with if statement to transfer control to statement when condition is false.
enum	It is used to declare enumerated data.
extern	It is used to declare extern storage class variable
float	Used for declaration of float type variable
for	Used for repetitive execution of statements
goto	It is used to transfer control from one statement to another
if	It is used for condition checking
int	Used for declaration of integer type variable
long	Used for declaration of long type variable
register	It is used to declare register storage class variable
return	It is used to return value from function.
short	Used for declaration of short type variable
signed	Used for declaration of signed type variable
sizeof	It returns memory size allocated to variable or data type
static	It is used to declare static storage class variable
struct	It is used to declare user defined data type structure

Any four keywords 1M

Use 1M

switch	It is used to make decision from multiple number of inputs
typedef	Used to redefine the name of an existing variable type.
union	It is used to declare the data type union
unsigned	Used for declaration of unsigned type variable
void	Specify that function does not return any value
volatile	It is used to declare a volatile variable
while	Used for repetitive execution of statements

(c) **Write the syntax of switch case statement.** **2M**

Ans.

```
switch(variable)
{
case value1:
statements
break;
case value2:
statements;
break;
.
.
.
default:
statements;
break;
}
```

Correct syntax 2M

(d) **State any two differences between while and do-while statement.** **2M**

(Note: Any 2 points shall be considered).

Ans.

while	Do-while
In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
The iterations do not occur if, the condition at the first iteration, appears false.	The iteration occurs at least once even if the condition is false at the first iteration.
It is an entry controlled loop	It is an exit controlled loop
while(condition) { body	do { body

Any two differences 1M each

}	}while(condition);

(e) **State difference between array and string.** **2M**

(Note: Any two valid points shall be considered).

Ans.

Array	String
Array can be of any type like int, float, char.	String can contain only characters.
Element Elements in an array can be accessed using its position like a[2].s in an array can be accessed using its position like a[2].	Characters in string are accessed sequentially from first to last.
Array does not end with a null character	String is ended with a '\0' character.
Array size once declared cannot be changed	String size can be modified using pointer.

Any two points 1M for each

(f) **Declare a structure student with element roll-no and name.** **2M**

Ans.

```
struct student
{
int roll_no;
char name[20];
};
```

Correct declaration 2M

(g) **Distinguish between call by value and call by reference.** **2M**

(Note: Any two points shall be considered).

Ans.

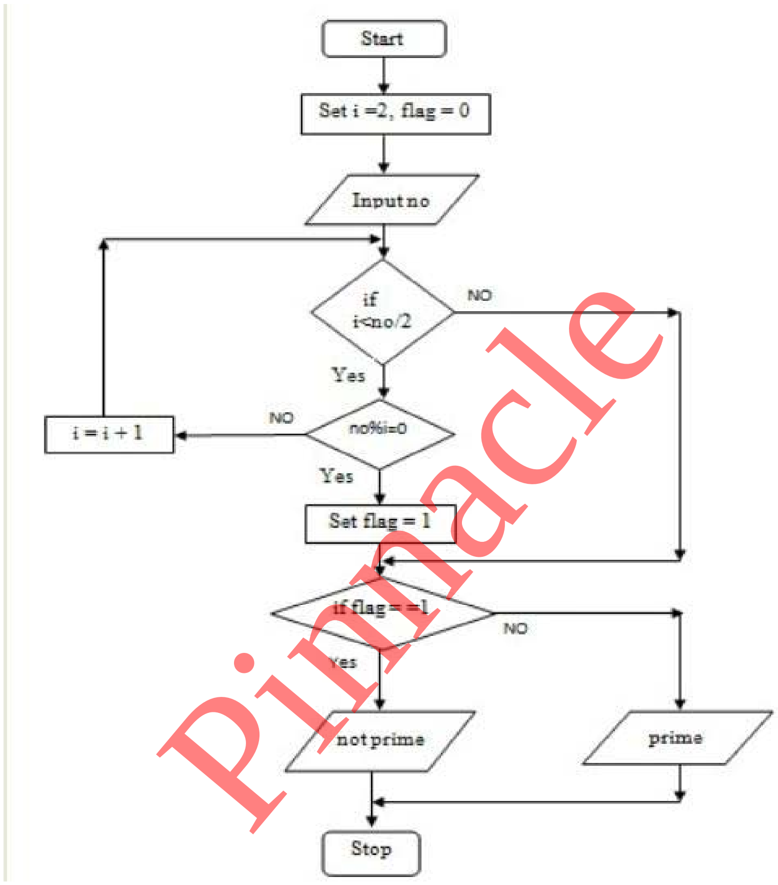
Call by value	Call by reference
A copy of actual arguments is passed to respective formal arguments.	The address of actual arguments is passed to formal arguments
Actual arguments will remain safe, they cannot be modified accidentally.	Alteration to actual arguments is possible within from called function; therefore the code must handle arguments carefully else you get unexpected results.
Address of the actual and formal arguments are different	Address of the actual and formal arguments are the same
Changes made inside the function is not reflected in other functions	Changes made in the function is reflected outside also.

Any two points 1M each

2. **Attempt any THREE of the following:** **12**

(a) **State four arithmetic operations perform on pointer with** **4M**

	Ans.	**example.** ***(Note: Code snippet shall be considered)*** The pointer arithmetic is done as per the data type of the pointer. The basic operations on pointers are **Increment:** It is used to increment the pointer. Each time a pointer is incremented, it points to the next location with respect to memory size . Example, If ptr is an integer pointer stored at address 1000,then ptr++ shows 1002 as incremented location for an int.It increments by two locations as it requires two bytes storage. **Decrement:** It is used to decrement the pointer. Each time a pointer is decremented, it points to the previous location with respect to memory size. Example, If the current position of pointer is 1002, then decrement operation ptr-- results in the pointer pointing to the location 1000 in case of integer pointer as it require two bytes storage. **Addition** When addition operation is performed on pointer, it gives the location incremented by the added value according to data type. Eg: If ptr is an integer pointer stored at address 1000, Then ptr+2 shows 1000+(2*2) = 1004 as incremented location for an int. **Subtraction** When subtraction operation is performed on the pointer variable, it gives the location decremented by the subtracted value according to data type. Eg: If ptr is an integer pointer stored at address 1004, Then ptr-2 shows 1004-(2*2) = 1000 as decremented location for an int.	***Each operatio n with example 1M***
	(b)	**Draw flowchart for checking weather given number is prime or**	**4M**

	Ans.	**not.**  Start Set i =2, flag = 0 Input no if i<no/2 NO Yes i = i + 1 NO no%i=0 Yes Set flag = 1 if flag = =1 NO Yes not prime prime Stop	*Correct logic 2M* *Symbols 2M*

	(c)
	Ans.
		Write a program to reverse the number 1234 (i.e. 4321) using function.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void findReverse();
		void main()
		{
		findReverse();
		}
		void findReverse()
		{
			4M
			Correct syntax 2M
			Correct logic 2M

```
int num, res=0,ans=0;
clrscr();
printf("Enter the number");
scanf("%d", &num);
while(num!=0)
{
res=num%10;
ans=ans*10+res;
num=num/10;
}
printf("Reverse number is %d", ans);
getch();
}
```

(d) **Differentiate between character array and integer array with respect to size and initialisation.** 4M

Ans.

Parameter	Character Array	Integer Array
Size	Last location in character array is filled with '\0' so the array size should be so declared that it should have one last location for '\0' character.	No extra location than the number of elements is required.

Initialization	Initialization can be done like :
	char str[4]={'a','b','c','\0'};
	char str[4]="abc";
		Initialization can be done like :
		int arr[4]={1,2,3,4};

Each parameter 2M

3. **Attempt any THREE of the following:** 12

(a) **Write a program to sum all the odd numbers between 1 to 20.** 4M

(Note: Any other correct logic shall be considered).

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int sum=0,i;
        clrscr();
        for(i=1;i<=20;i++)
```

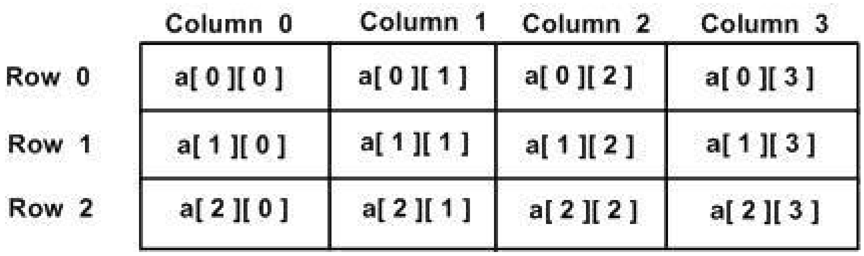
Correct logic 2M

		{
		if(i%2==1)
		sum=sum+i;
		}
		printf(“sum of odd no‟s between 1 to 20 is %d”,sum);
		getch();
		}
			Correct syntax 2M

	(b) **Ans.**	**Explain any four bit-wise operator used in ‘C’ with example.** **Bitwise operators:** **Bitwise OR – |** It takes 2 bit patterns and performs OR operations on each pair of corresponding bits. The following example will explain it. 1010 1100 -------- OR 1110 **Bitwise AND – &** It takes 2 bit patterns and performs AND operations with it. 1010 1100 ------- AND 1000 ------- The Bitwise AND will take pair of bits from each position, and if only both the bit is 1, the result on that position will be 1. Bitwise AND is used to Turn-Off bits. **Bitwise NOT** One’s complement operator (Bitwise NOT) is used to convert each “1-bit to 0-bit” and “0-bit to1-bit”, in the given binary pattern. It is a unary operator i.e. it takes only one operand. 1001 NOT 0110 ------- **Bitwise XOR ^** Bitwise XOR ^, takes 2 bit patterns and perform XOR operation with it.	**4M** ***Explanation with example of any four bitwise operator 1M each***

		0101
		0110
		------ 	
		XOR 0011
		------ 	
		
		Left shift Operator – <<
		The left shift operator will shift the bits towards left for the given number of times.
		int a=2<<1;
		
		Right shift Operator – >>
		The right shift operator will shift the bits towards right for the given number of times.
		int a=8>>1;	

	(c) **Ans.**	**With suitable example, explain how two dimensional arrays can be created.** The array which is used to represent and store data in a tabular form is called as two dimensional array. Such type of array is specially used to represent data in a matrix form. Declaration of two dimensional arrays: *Syntax:-* Data type arrayname [row size] [column size]; *Eg:* int arr[3][4]; This will declare array “arr” with 3 rows and 4 columns. A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array a, which contains three rows and four columns can be shown as follows –	**4M** ***Explanation 2M***



	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]

Thus, every element in the array a is identified by an element name of the form a[i][j], where 'a' is the name of the array, and 'i' and 'j' are the subscripts that uniquely identify each element in 'a'.

Example :

```
main()
{
        int a[2][2]={{1,2},{4,5}};
        int i,j;
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                        printf( "%d",a[i][j]);
                }
                printf("\n");
        }
}
```

Example 2M

(d) Ans. **Explain any two string functions with example.** 4M

Strlen function:

strlen( ) function in C gives the length of the given string. strlen( ) function counts the number of characters in a given string and returns the integer value. It stops counting the character when null character is found. Because, null character indicates the end of the string in C.

Syntax:

strlen(stringname);

Example:

Consider str1="abc"

strlen(str1); returns length of str1 as 3

Explanation of any two string functions 1M each, example 1M each

strcat() function:

In C programming, strcat() concatenates (joins) two strings. It concatenates source string at the end of destination string.

Syntax:

strcat(destinationsource, source string);

Example:

Consider str1=”abc” and str2=”def”

strcat(str1,str2); returns abcdef in str1 and str2 remains unchanged.

strcpy() function

strncpy( ) function copies portion of contents of one string into another string.

Syntax:

strncpy(destination string, source string, size);

Example:

Consider str1=”abc”

strcpy(str1,str2); returns abcstr2

strcmp() function

The strcmp function compares two strings which are passed as arguments to it. If the
strings are equal then function returns value 0 and if they are not equal the function
returns some numeric value.

Syntax:

strcmp(str1, str2);

Example:

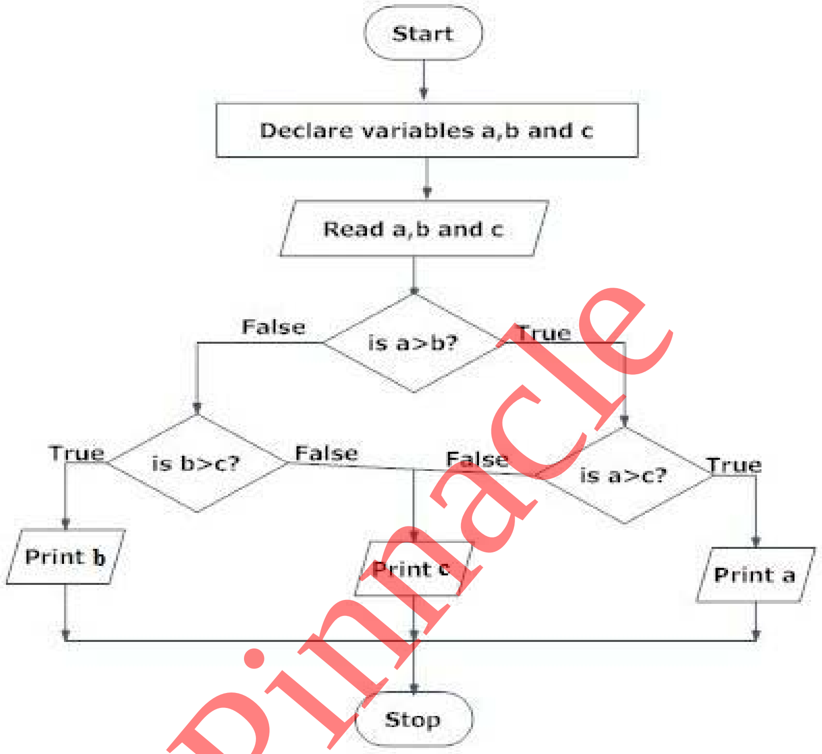
Consider str1=”abc” and str2=”abc”

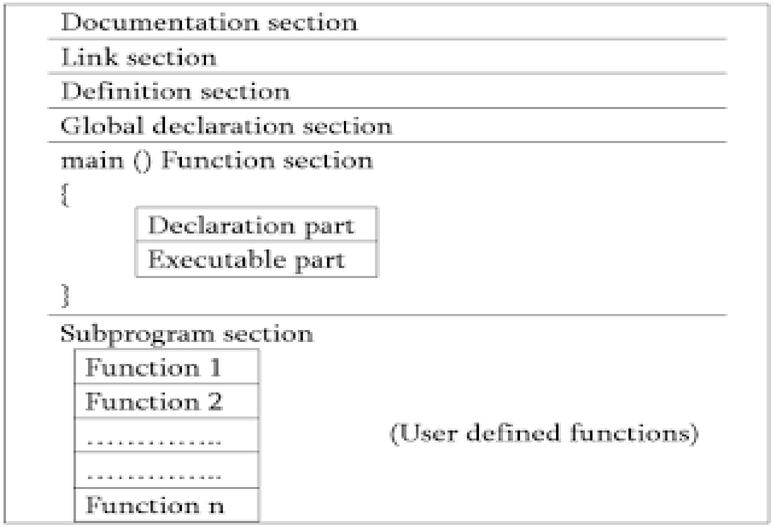
Then strcmp(str1,str2) returns 0 as both the strings are same.

4. **Attempt any THREE of the following:** **12**

(a) **Draw flowchart for finding largest number among three numbers.** **4M**

Ans.

		 Start Declare variables a,b and c Read a,b and c is a>b? False True is b>c? True False is a>c? False True Print b Print c Print a Stop	*Correct flowchart 4M*

	(b)
	Ans.
		Describe generic structure of 'C' program.
		
		Documentation section
		Link section
		Definition section
		Global declaration section
		main () Function section
		{
		Declaration part
		Executable part
		}
		Subprogram section
		Function 1
		Function 2
		…………..
		…………..
		Function n
		(User defined functions)
			4M
			List of sections from structure 1M

		Documentation section: The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later. **Link section**: The link section provides instructions to the compiler to link functions from the system library such as using the #include directive. **Definition section**: The definition section defines all symbolic constants such using the #define directive. **Global declaration section**: There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. **Declaration part**: The declaration part declares all the variables used in the executable part. **Subprogram section**: If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order. **Header files** A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. **Include Syntax** Both the user and the system header files are included using the preprocessing directive #include. **‘main’ function** **main()** function is the entry point of any C program. It is the point at which execution of program is started. Every C program have a main() function.	***Correct description of structure 3M***

	(c)
	
	Ans.
		Write a program to take input as a number and reverse it by while loop.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int no;
		int sum=0,rem;
		printf("\n Enter number:");
		scanf("%d",&no);
		while(no>0)
		{
		rem=no%10;
		no=no/10;
		sum=sum*10+rem;
		}
		printf("\nsum=%d",sum);
		getch();
		}
			4M
			
			Accept input 1M
			
			Use of while loop 1M
			
			correct syntax 2M

	(d)
	
	Ans.
		Write a program to accept 10 numbers in array and arrange them in ascending order.
		(Note: Any other correct logic shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int arr[10],i,j,temp;
		clrscr();
		printf("Enter array elements:");
		for(i=0;i<10;i++)
		{
		scanf("%d",&arr[i]);
		}
		printf("\n\n Array elements are:");
		for(i=0;i<10;i++)
		{
		printf("%d ",arr[i]);
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

```
    }
    for(j=0;j<10;j++)
    {
    for(i=0;i<10;i++)
    {
        if(arr[i+1]<arr[i])
        {
                temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
        }
    }
}
printf("\n\nArray elements in ascending order are:");
for(i=0;i<10;i++)
{
     printf("%d ",arr[i]);
}
getch();
}
```

	(e)
	
	Ans.
		Explain meaning of following statement with reference to pointers:
		int *a, b;
		b=20;
		***a=b;**
		A=&b;
		
		int *a,b;
		It is declaration of integer pointer a and integer variable b
		
		b=20;
		value 20 is assigned to variable b.
		
		***a=b;**
		Value of b is assigned to pointer a.
		
		A=&b;
		Address of b is assigned to variable A.
			4M
			
			Correct meaning of each statement 1M

5.		**Attempt any TWO of the following:**	**12**
	(a)	**Write a program to perform addition, subtraction, multiplication and division of two integer number using function.** ***(Note: Any other correct logic shall be considered).***	**6M**

	Ans.	#include<stdio.h>
		#include<conio.h>
		void add(int x,int y)
		{
		printf("\nAddition=%d",x+y);
		}
			Add function 1M

		void sub(int x,int y)
		{
		printf("\nSubtraction=%d",x-y);
		}
			sub function 1M

		void mult(int x,int y)
		{
		printf("\nMultiplication=%d",x*y);
		}
			Mult function 1M

		void div(int x,int y)
		{
		printf("\nDivision=%d",x/y);
		}
			Div function 1M

		void main()
		{
		intx,y;
		clrscr();
		printf("Enter x:");
		scanf("%d",&x);
		printf("Enter y:");
		scanf("%d",&y);
		add(x,y);
		sub(x,y);
		mult(x,y);
		div(x,y);
		getch();
		}
			Main function 2M

	(b)	**Define Array. Write a program to accept ten numbers in array. Sort array element and display.**	**6M**

Ans.

Definition of Array:
An **array** is a collection of data elements, all of the same type, accessed using a common name.

Array definition 1M

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10],i,j,temp;
clrscr();
printf("Enter numbers:");
for(i=0;i<10;i++)
scanf("%d",&a[i]);
for(i=0;i<10;i++)
{
for(j=i+1;j<10;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
printf("\n Sorted array elements:");
for(i=0;i<10;i++)
printf("\n %d",a[i]);
getch();
}
```

Accepting array 1M

Sorting logic 3M

Display sorted array 1M

(c) **Write a program to print reverse of a entered string using pointer.** 4M

(Note: Any other correct logic shall be considered).

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

		char str[10],*ptr;
		int l=0;
		clrscr();
		printf("Enter string:");
		scanf("%s",str);
		ptr=str;
		while(*ptr!='\0')
		{
		l=l+1;
		ptr=ptr+1;
		}
		while(l>0)
		{
		ptr=ptr-1;
		printf("%c",*ptr);
		l=l-1;
		}
		getch();
		}
			Accepting string 1M
			pointer initialization1M
			logic of reverse using pointer 3M
			Displaying reverse string 1M

6.		**Attempt any TWO of the following:**	**12**
	(a)	**Explain recursion with suitable example. List any two advantages.**	**6M**

	Ans.	Recursion means a function calls itself repetitively. A recursive function contains a function call to itself inside its body.
		
		Example:
		#include<stdio.h>
		#include<conio.h>
		int factorial(int N);
		void main()
		{
		int N,fact;
		clrscr();
		printf("Enter number:");
		scanf("%d",&N);
		fact=factorial(N);
			Explanation of recursion 1M
			Example 3M

		printf("\n Factorial is:%d",fact);
		getch();
		}
		int factorial(int N)
		{
		if(N==1)
		return(1);
		else
		return(N*factorial(N-1));
		}
		
		Advantages:
		• Reduces length of the program
		• Reduces unnecessary calling of a function.
		• Useful when same solution is to be applied many times.
			Any two Advantages 2M

	(b)
	
	Ans.
		Write a program to accept ten numbers and print average of it.
		(Note: Program without array shall be considered).
		#include<stdio.h>
		#include<conio.h>
		void main()
		{
		int a[10],i,sum=0;
		float avg;
		clrscr();
		printf("Enter numbers:");
		for(i=0;i<10;i++)
		scanf("%d",&a[i]);
		for(i=0;i<10;i++)
		sum=sum+a[i];
		avg=sum/10;
		printf("\n Average =%f", avg);
		getch();
		}
			6M
			
			Accepting 10 numbers 2M
			
			Calculating average 2M
			
			Displaying average 2M

	(c) **Ans.**	**Enlist different format specifiers with its use.** Format specifier tells the compiler what type of data a variable holds during taking input and printing output using scanf() and printf() functions respectively. Format specifiers used in C programming:	**6M**

Format specifier	Use
%d	Specify data type as short signed
%u	Specify data type as short unsigned
%ld	Specify data type as long singed
%lu	Specify data type as long unsigned
%x	Specify data type as unsigned hexadecimal
%o	Specify data type as unsigned octal
%f	Specify data type as float
%lf	Specify data type as double
%Lf	Specify data type as long double
%c	Specify data type as signed character
%s	Specify data type as unsigned group of characters(Strings)

Any six format specifiers with use 1M each