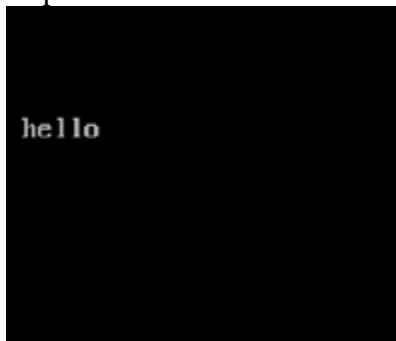


1. /* Write a program to print “HELLO”*/

```
#include<stdio.h>
void main()
{
clrscr();
printf("\n hello");
getch();
}
```

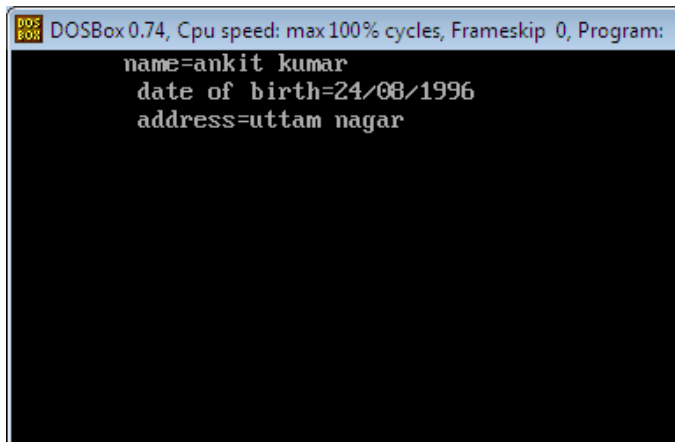
output:-



2. Write a program to display your name, date of birth and address?

```
#include <stdio.h>
#include <conio.h>
void main()
{
clrscr();
printf("name=ankit kumar");
printf("date of birth=24/08/1996");
printf("address=uttam nagar");
getch();
}
```

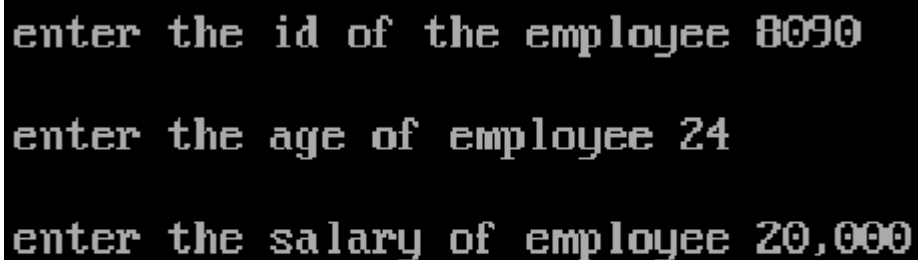
Screenshot:-

A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:". The main window area is black with white text displaying the output of a program: "name=ankit kumar", "date of birth=24/08/1996", and "address=uttam nagar".

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:
name=ankit kumar
date of birth=24/08/1996
address=uttam nagar
```

3. /* WAP to ask the user for employee id, age, salary and display the same*/

```
#include<stdio.h>
void main()
{
    int id,age,salary;
    clrscr();
    printf("\n enter the id of the employee");
    scanf("%d",&id);
    printf("\n enter the age of employee");
    scanf("%d",&age);
    printf("\n enter the salary of employee");
    scanf("%d",&salary);
    getch();
}
```

A screenshot of a DOSBox window showing the execution of the C program. It displays three prompts and their corresponding inputs: "enter the id of the employee 8090", "enter the age of employee 24", and "enter the salary of employee 20,000".

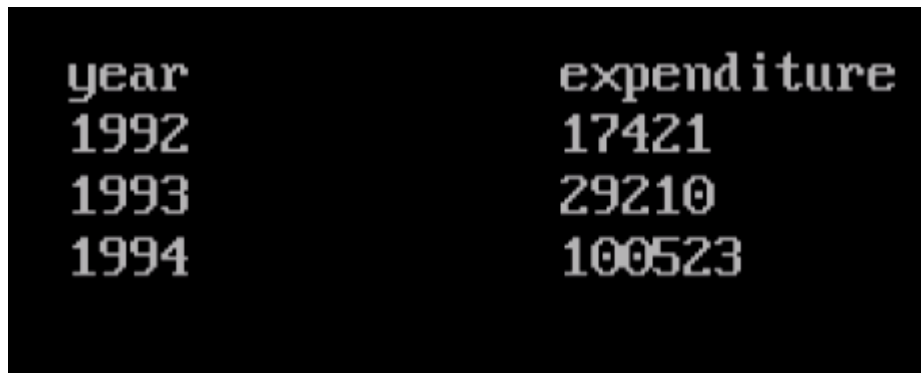
```
enter the id of the employee 8090
enter the age of employee 24
enter the salary of employee 20,000
```

output:-

4. :- /*WAP to generate the following table*/

YEAR	EXPENDITURE
1992	17421
1993	29210
1994	100523

```
#include <stdio.h>
void main()
{
    clrscr();
    printf("\n year    \t expenditure");
    printf("\n 1992    \t 17421");
    printf("\n 1993    \t 29210");
    printf("\n 1994    \t 100523");
    getch();
}
```



year	expenditure
1992	17421
1993	29210
1994	100523

Output:-

5. :- /*WAP to display the following*/

MATHS	90
PHYSICS	77
CHEMISTRY	69

```
#include <stdio.h>
void main()
{
    clrscr();
    printf("\n maths    \t 90");
    printf("\n physics    \t 77");
    printf("\n chemistry    \t 69");
    getch();
}
```

```
maths          90
physics        77
chemistry      69_
```

Output:-

6. :- /* WAP to calculate the area of circle*/

```
#include<stdio.h>
void main()
{
float radius,pie,area;
pie=3.14;
clrscr();
printf("\n enter the value of radius");
scanf("\n %f",&radius);
area=(pie*radius*radius);
printf("\n area of circle=%f",area);
getch();
}
```

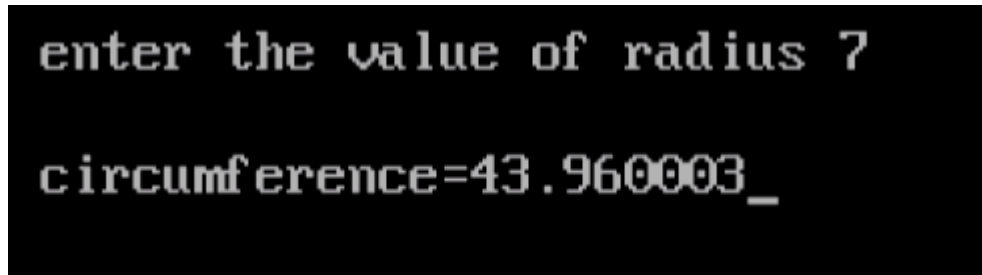
```
enter the value of radius 2
area of circle=12.560000
```

Output:-

7. :- /*WAP to calculate the circumference of a circle*/

```
#include<stdio.h>
void main()
{
float circumference,radius,pie;
pie=3.14;
clrscr();
printf("\n enter the value of radius");
scanf("\ %f",&radius);
circumference=(2*pie*radius);
printf("\n circumference=%f",circumference);
getch();
}
```

Output:-

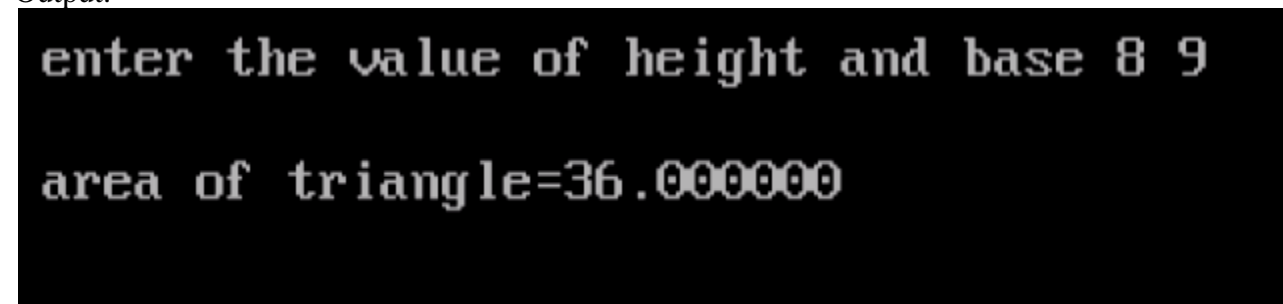
A screenshot of a terminal window with a black background and white text. The first line shows the prompt 'enter the value of radius ?' followed by the input '7'. The second line shows the output 'circumference=43.960003_'.

```
enter the value of radius ?
circumference=43.960003_
```

8. /*WAP to calculate the area of a triangle*/

```
#include<stdio.h>
void main()
{
float height,base,area;
clrscr();
printf("\n enter the value of height and base");
scanf("\n %f %f",&base,&height);
area=(base*height)/2;
printf("\n area of triangle=%f",area);
getch();
}
```

Output:-

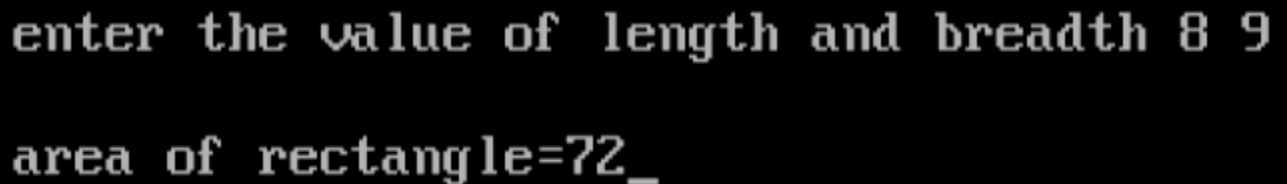
A screenshot of a terminal window with a black background and white text. The first line shows the prompt 'enter the value of height and base' followed by the inputs '8' and '9'. The second line shows the output 'area of triangle=36.000000'.

```
enter the value of height and base 8 9
area of triangle=36.000000
```

9. :- /*WAP to calculate area of rectangle*/

```
#include<stdio.h>
void main()
{
int length,breadth,area;
clrscr();
printf("\n enter the value of length and breadth");
scanf("\n %d %d",&length,&breadth);
area= (length*breadth);
printf("\n area of rectangle=%d",area);
getch();
}
```

Output:-



The screenshot shows the output of the program in a terminal window. The first line is "enter the value of length and breadth 8 9", where "8 9" are the user inputs for length and breadth. The second line is "area of rectangle=72_", where "72" is the calculated area and "_" is the prompt character from getch().

10. :- /* WAP to operate swapping of two number with using 3rd variable and without using 3rd variable*/

```
#include<stdio.h>
void main()
{
int a,b,c,choice;
clrscr();
printf("\n press 1 for using 3rd variable");
printf("\n press 2 for without using 3rd variable");
printf("\n enter your choice");
scanf("\n %d",&choice);
switch (choice)
{
case 1:
printf("\n enter the value of a and b");
scanf("\n %d %d",&a,&b);
c=a;
a=b;
b=c;
printf("\n after swapping value of a is=%d",a);
printf("\n after swapping value of b is=%d",b);
break;
```

Output:-	1:-	using	3 rd	variable
----------	-----	-------	-----------------	----------

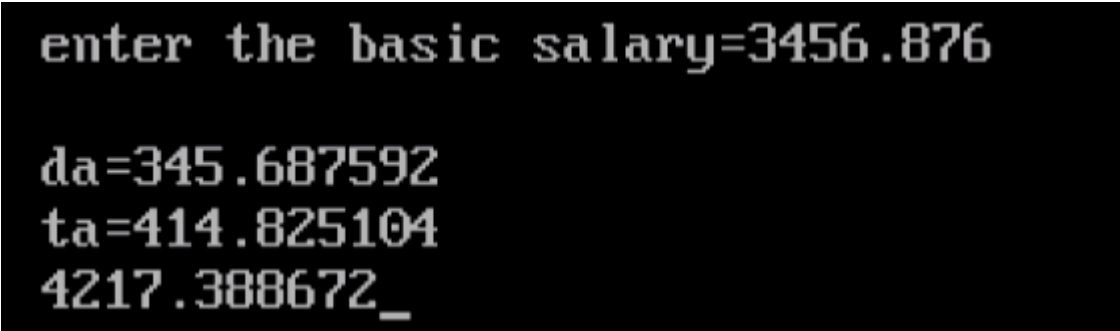
2:- without using 3rd variable.

7

11. :- /* WAP to calculate total salary*/

```
#include<stdio.h>
void main()
{
float bs,da,ta,total;
clrscr();
printf("\n enter the basic salary=");
scanf("\n basic salary=%f",&bs);
da=(bs*10)/100;
printf("\n da=%f",da);
ta=(bs*12)/100;
printf("\n ta=%f",ta);
total=bs+da+ta;
printf("\n %f",total);
getch();
}
```

Output:-

A screenshot of a terminal window with a black background and white text. The output shows the program's execution results for a basic salary of 3456.876. The displayed values are: da=345.687592, ta=414.825104, and a total of 4217.388672 followed by a cursor underscore.

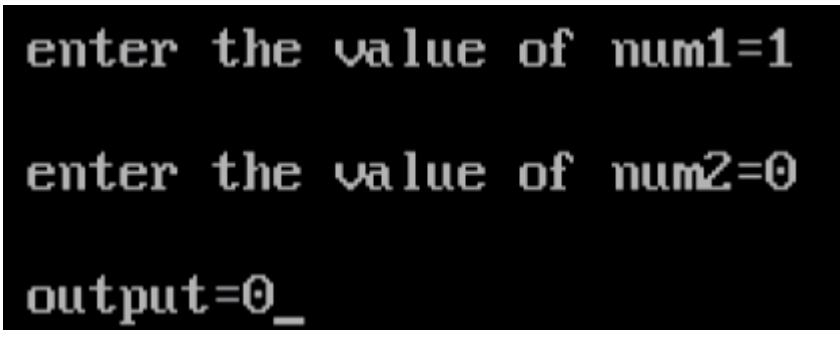
```
enter the basic salary=3456.876

da=345.687592
ta=414.825104
4217.388672_
```


12. :- /*WAP to use bitwise operator between two integers*/

```
#include<stdio.h>
void main()
{
int num1,num2,output;
clrscr();
printf("\n enter the value of num1=");
scanf("\n %d",&num1);
printf("\n enter the value of num2=");
scanf("\n %d",&num2);
output= (num1 & num2);
printf("\n output=%d",output);
getch();
}
```

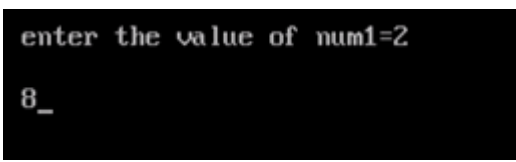
Ooutput:-



13. :- /*WAP to shift input data by 2 bits to the left*/

```
#include<stdio.h>
void main()
{
int num1,num2;
clrscr();
printf("\n enter the value of num1=");
scanf("\n %d",&num1);
num2=num1<<2;
printf("\n %d",num2);
getch();
}
```

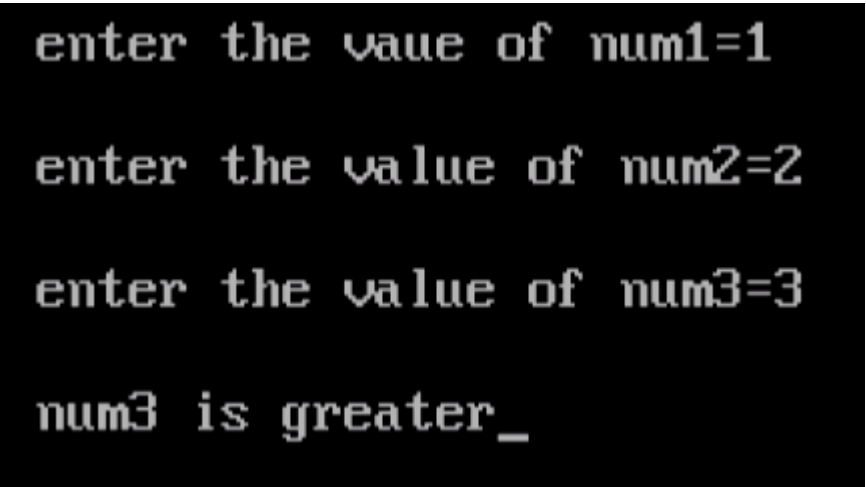
Output:-



14. :- /*WAP to find greatest in three numbers*/

```
#include<stdio.h>
void main()
{
int num1,num2,num3;
clrscr();
printf("\n enter the vaue of num1=");
scanf("%d",&num1);
printf("\n enter the value of num2=");
scanf("%d",&num2);
printf("\n enter the value of num3=");
scanf("\n %d",&num3);
if ((num1>num2) && (num1>num3))
{
printf("\ num1 is greater than num2 and num3");
}
else if (num3>num2)
{
printf("\n num3 is greater");
}
else
{
printf("\n num2 is greater");
}
getch();
}
```

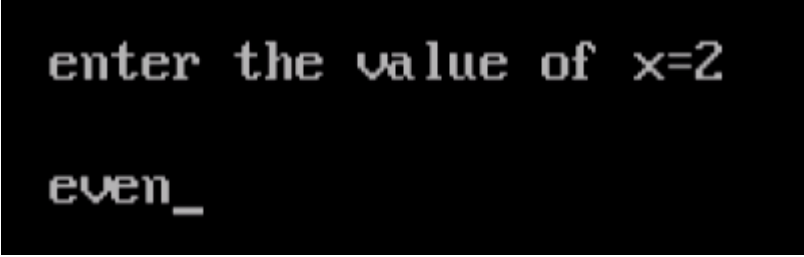
Output:-

A screenshot of a terminal window with a black background and white text. It shows the output of the C program: 'enter the vaue of num1=1', 'enter the value of num2=2', 'enter the value of num3=3', and 'num3 is greater_'.

```
enter the vaue of num1=1
enter the value of num2=2
enter the value of num3=3
num3 is greater_
```

15. :- /*WAP to show the use of conditional operator*/

```
#include<stdio.h>
void main()
{
int x;
clrscr();
printf("\n enter the value of x=");
scanf("\n %d",&x);
(x%2==0?printf("\n even"):printf("\n odd"));
getch();
}
```

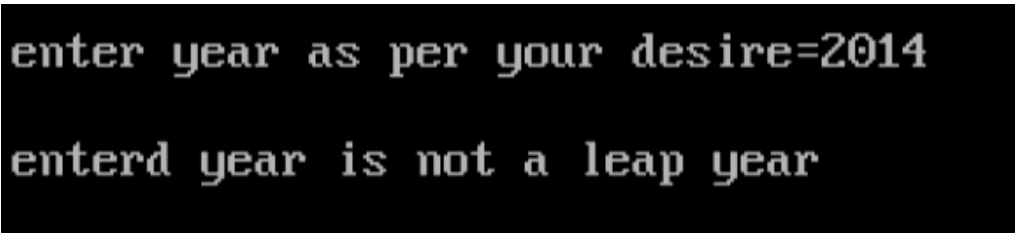


enter the value of x=2
even_

Output:-

16. :- /*WAP to find that entered year is a leap year or not*/

```
#include<stdio.h>
void main()
{
int x;
clrscr();
printf("\n enter year as per your desire=");
scanf("%d",&x);
if (x%4==0)
{
printf("\n entered year is a leap year");
}
else
{
printf("\n entered year is not a leap year");
}
getch();
}
```

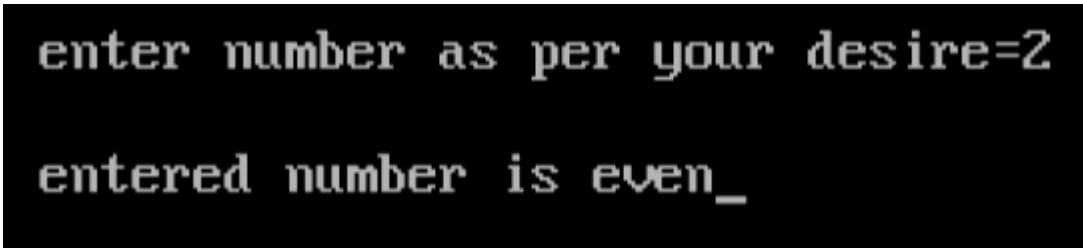


enter year as per your desire=2014
entered year is not a leap year

Output:-

17. :- /* WAP to check whether the entered number is even or odd*/

```
#include<stdio.h>
void main()
{
int x;
clrscr();
printf("\n enter number as per your desire=");
scanf("%d",&x);
if (x%2==0)
{
printf("\n entered number is even");
}
else
{
printf("\n entered year is odd");
}
getch();
}
```



enter number as per your desire=2
entered number is even_

Output:-

18. :- /*WAP to display week days from Monday to Sunday*/

```
#include<stdio.h>
void main()
{
int choice;
clrscr();
printf("\n press 1 for monday");
printf("\n press 2 for tuesday");
printf("\n press 3 for wednesday");
printf("\n press 4 for thursday");
printf("\n press 5 for friday");
printf("\n press 6 for saturday");
printf("\n press 7 for sunday");
printf("\n enter your choice=");
scanf("\n %d",&choice);
switch (choice)
{
case 1:
printf("\n monday");
```

```
//break;
case 2:
printf("\n tuesday");
//break;
case 3:
printf("\n wednesday");
//break;
case 4:
printf("\n thursday");
//break;
case 5:
printf("\n friday");
//break;
case 6:
printf("\n saturday");
//break;
case 7:
printf("\n sunday");
break;
default:
printf("\n choice does not match");
}
getch();
}
```

```
press 1 for monday
press 2 for tuesday
press 3 for wednesday
press 4 for thursday
press 5 for friday
press 6 for saturday
press 7 for sunday
enter your choice=1

monday
tuesday
wednesday
thursday
friday
saturday
sunday
```

Output:-

19. :- /*WAP to perform arithmetic operation using switch case*/

```
#include<stdio.h>
void main()
{
int num1,num2,addition,substraction,multiplication,mod,division,choice;
char s;
clrscr();
printf("\n enter the value of num1=");
scanf("\n %d",&num1);
printf("\n enter the value of num2=");
scanf("\n %d",&num2);
printf("\n enter 1 for addition:");
printf("\n enter 2 for subtraction:");
printf("\n enter 3 for multiplication:");
printf("\n enter 4 for division:");
printf("\n enter 5 for mod:");
printf("\n enter your choice=");
```

```
scanf("\n %d",&choice);
switch (choice)
{
case 1:
addition=num1+num2;
printf("\n addition= %d+%d=%d",num1,num2,addition);
break;
case 2:
subtraction=num1-num2;
printf("\n subtraction=%d-%d=%d",num1,num2,subtraction);
break;
case 3:
multiplication=num1*num2;
printf("\n multiplication=%d*%d=%d",num1,num2,multiplication);
break;
case 4:
division=num1/num2;
printf("\n division=%d/%d=%d",num1,num2,division);
break;
case 5:
mod=num1%num2;
printf("\n mod=%d%d=%d",num1,num2,mod);
break;
default:
printf("\n no operation can perform");
}
getch();
}
```

```
enter the value of num1=2
enter the value of num2=3

enter 1 for addition:
enter 2 for subtraction:
enter 3 for multiplication:
enter 4 for division:
enter 5 for mod:
enter your choice=1

addition= 2+3=5
```

Output:

20. :- /*WAP to display 10 natural nos*/

```
#include<stdio.h>
void main()
{
int x;
clrscr();
x=1;
while (x<=10)
{
printf("\n counting from 1 to 10=%d",x);
x++;
}
getch();
}
```



```
counting from 1 to 10=1
counting from 1 to 10=2
counting from 1 to 10=3
counting from 1 to 10=4
counting from 1 to 10=5
counting from 1 to 10=6
counting from 1 to 10=7
counting from 1 to 10=8
counting from 1 to 10=9
counting from 1 to 10=10
```

Output:-

21. :- /*WAP to print 1st 10 even number*/

```
#include<stdio.h>
void main()
{
int x;
clrscr();
x=1;
while (x<=10)
{
if (x%2==0)
{
printf("\n even number between 1 to 10=%d",x);
}
x++;
}
getch();
}
```

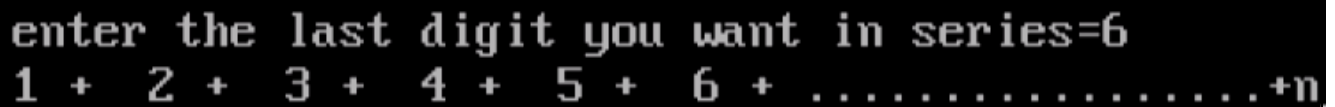
```
even number between 1 to 10=2
even number between 1 to 10=4
even number between 1 to 10=6
even number between 1 to 10=8
even number between 1 to 10=10
```

Output:-

22. :- /*WAP to print following series*/
1:- 1+2+3+.....+n

```
#include<stdio.h>
void main()
{
int num1=1,n;
clrscr();
printf("\n enter the last digit you want in series");
scanf("\ %d",&n);
do
{
printf("%d+ ",num1);
num1++;
}
while (num1<=n);
printf(".....+n");
getch();
}
```

Output:-



enter the last digit you want in series=6
1 + 2 + 3 + 4 + 5 + 6 ++n

/*WAP to print series*/
2:- 1+3+5+7+.....+n

```
#include<stdio.h>
void main()
{
int num1=1,n;
clrscr();
printf("\n enter the last digit you want in series=");
scanf("\ %d",&n);
do
{
printf(" %d + ",num1);
num1=num1+2;
}
while (num1<=n);
printf(".....+n");
getch();
}
```

Output:-

```
enter the last digit you want in series=7
1 + 3 + 5 + 7 + .....+n
```

/*WAP to print series*/

3:- $n+(n-1)+(n-2)+\dots+1$

```
#include<stdio.h>
void main()
{
int last=0;
char n='n';
clrscr();
do
{
printf("%c+",n);
last++;
printf("(%c-%d)+",n,last);
} while (n<=1);
printf(".....+1");
getch();
}
```

```
n+(n-1)+.....+1_
```

Output:-

23. Program to reverse a given number

```
#include <stdio.h>
```

```
int main()
{
int n, reverse = 0;

printf("Enter a number to reverse\n");
scanf("%d", &n);

while (n != 0)
{
reverse = reverse * 10;
reverse = reverse + n%10;
n      = n/10;
}
```

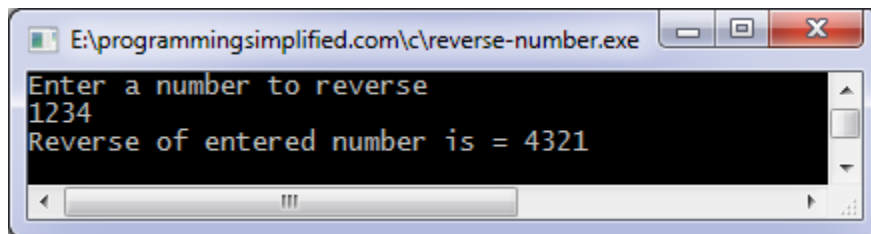
```

}

printf("Reverse of entered number is = %d\n", reverse);

return 0;
}

```



24. Program to print a table of any number.

```

/* C program to find multiplication table up to 10. */
#include <stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer to find multiplication table: ");
    scanf("%d",&n);
    for(i=1;i<=10;++i)
    {
        printf("%d * %d = %d\n", n, i, n*i);
    }
    return 0;
}

```

Output

Enter an integer to find multiplication table: 9

```

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

```

25 Program to print Fibonacci series

0 1 1 2 3 5.....n

```
/* Fibonacci Series c language */
#include<stdio.h>

int main()
{
    int n, first = 0, second = 1, next, c;

    printf("Enter the number of terms\n");
    scanf("%d",&n);

    printf("First %d terms of Fibonacci series are :-\n",n);

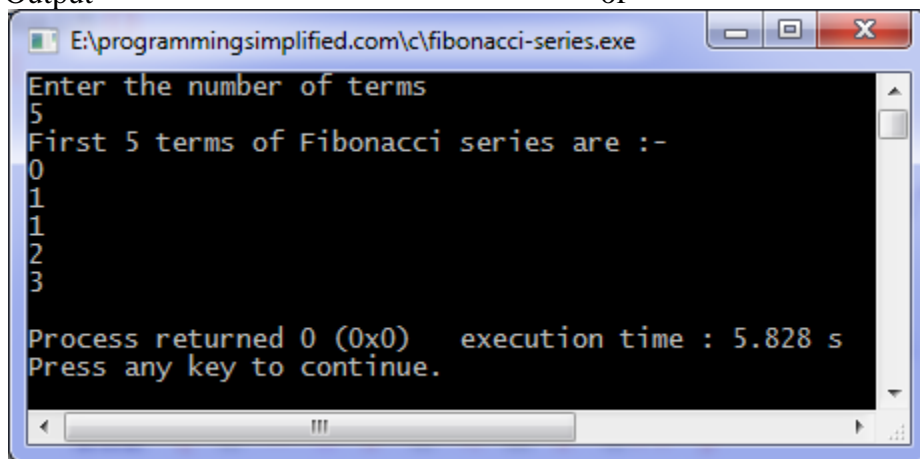
    for ( c = 0 ; c < n ; c++ )
    {
        if ( c <= 1 )
            next = c;
        else
        {
            next = first + second;
            first = second;
            second = next;
        }
        printf("%d\n",next);
    }

    return 0;
}
```

Output

of

program:



```
E:\programmingsimplified.com\c\fibonacci-series.exe
Enter the number of terms
5
First 5 terms of Fibonacci series are :-
0
1
1
2
3
Process returned 0 (0x0)   execution time : 5.828 s
Press any key to continue.
```

26 Program to find factorial of a number

```
#include <stdio.h>

int main()
{
    int c, n, fact = 1;

    printf("Enter a number to calculate it's factorial\n");
    scanf("%d", &n);

    for (c = 1; c <= n; c++)
        fact = fact * c;

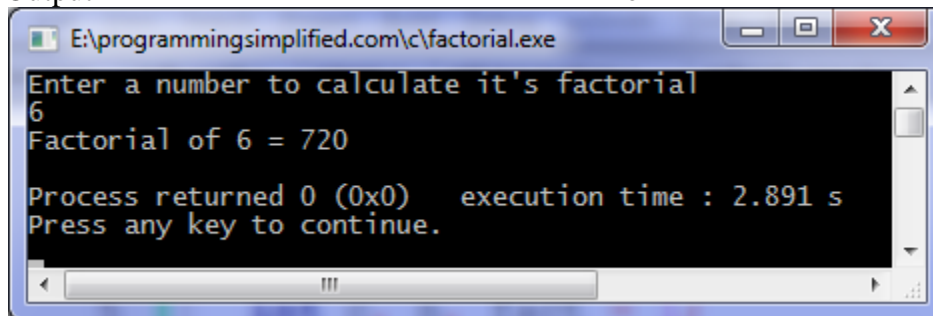
    printf("Factorial of %d = %d\n", n, fact);

    return 0;
}
```

Output

of

code:



27 Program to find whether given no is a prime no or not.

/* C program to check whether a number is prime or not. */

```
#include <stdio.h>
int main()
{
    int n, i, flag=0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2;i<=n/2;++i)
    {
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }
}
```

```

    }
}
if (flag==0)
    printf("%d is a prime number.",n);
else
    printf("%d is not a prime number.",n);
return 0;
}

```

Output

Enter a positive integer: 29
29 is a prime number.

28 program to display sum of series $1 + 1/2 + 1/3 + \dots + 1/n$

```

#include <stdio.h>

void main()
{
    double number, sum = 0, i;

    printf("\n enter the number ");
    scanf("%lf", &number);
    for (i = 1; i <= number; i++)
    {
        sum = sum + (1 / i);
        if (i == 1)
            printf("\n 1 +");
        else if (i == number)
            printf(" (1 / %lf)", i);
        else
            printf(" (1 / %lf) + ", i);
    }
    printf("\n The sum of the given series is %.2lf", sum);
}

```

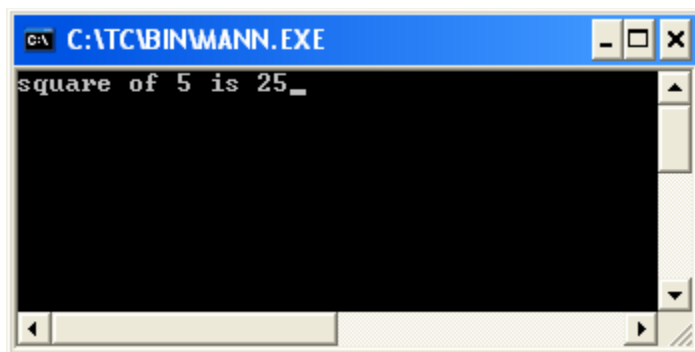
Output:

enter the number 4

$1 + (1/2.000000) + (1/3.000000) + (1/4.000000)$
The sum of the given series is 2.08

29 Program to find square of a number using functions

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5,b,c;
clrscr();
b=square(a);
printf("square of %d is %d",a,b);
getch();
}
square (a)
{
int c;
c=a*a;
return(c);
//getch();
}
```



30 Program to swap two numbers using functions(call by value)

ANS.

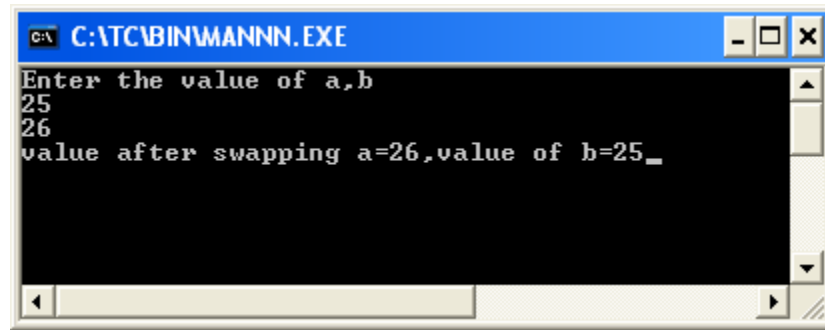
```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
swap();
getch();
}
swap()
{
int a,b;
printf("Enter the value of a,b");
scanf("%d %d",&a,&b);
```



```

a=a+b;
b=a-b;
a=a-b;
printf("value after swapping a=%d,value of b=%d",a,b);
return;
}

```



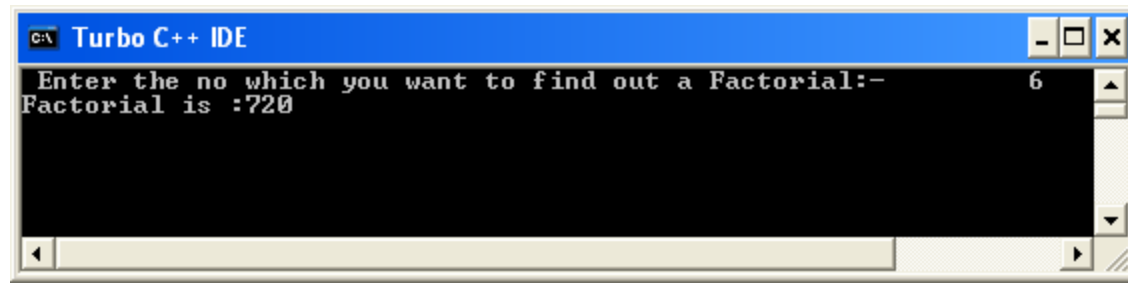
31 Program to find factorial of a number using functions.

ANS.

```

#include<stdio.h>
#include<conio.h>
void main()
{
int fact;
clrscr();
Fact();
getch();
}
Fact()
{
int a,f=1,i;
printf(" Enter the no which you want to find out a Factorial:- \t");
scanf("%d", &a);
for(i=1;i<=a;i++)
{
f=f*i;
}
printf("Factorial is :%d", f);
return;
}

```

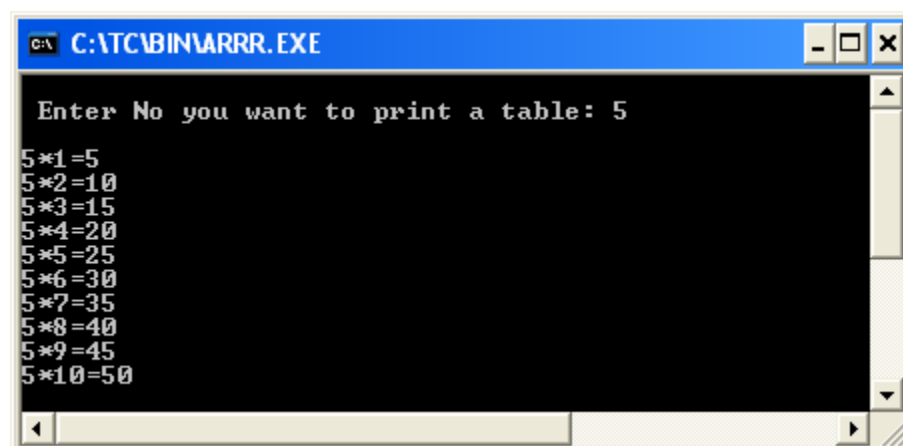


32 Program to show table of a number using functions.

ANS.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
table();

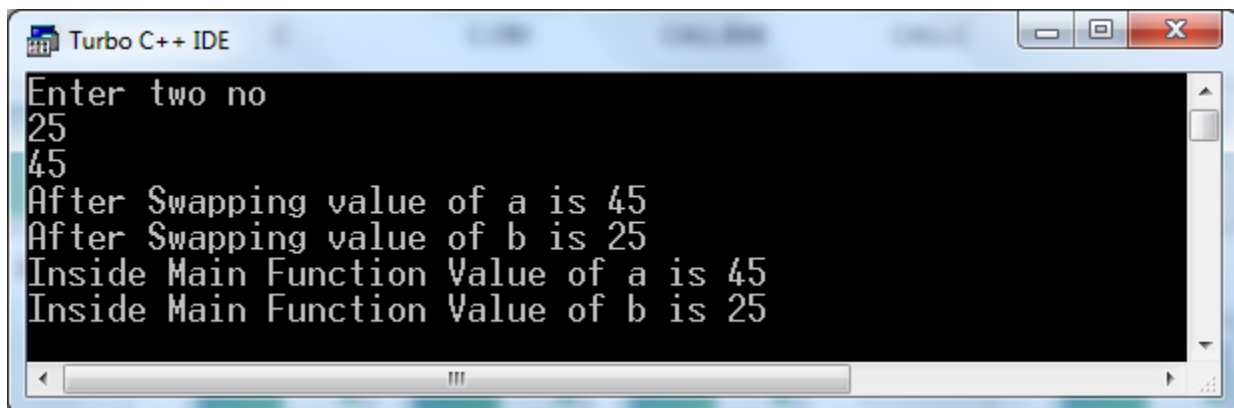
getch();
}
table()
{
int a,b,c;
printf("\n Enter No you want to print a table:");
scanf("\n%d",&a);
for(b=1;b<=10;b++)
{
c=b*a;
printf("\n%d*%d=%d",a,b,c);
}
return;
}
```



33 Program to swap two numbers using call by reference.

ANS.

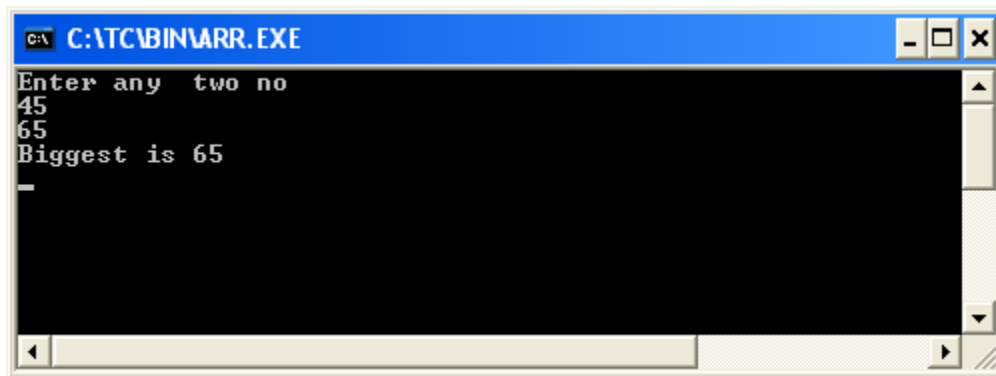
```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b,Swap(int*,int*);
    clrscr();
    printf("Enter two no \n");
    scanf("%d %d",&a,&b);
    Swap(&a,&b);
    printf("Inside Main Function Value of a is %d \n",a);
    printf("Inside Main Function Value of b is %d \n",b);
    getch();
}
Swap(int*x, int*y)
{
    int *temp;
    *temp=*x;
    *x=*y;
    *y=*temp;
    printf("After Swapping value of a is %d \n",*x);
    printf("After Swapping value of b is %d \n",*y);
}
```

A screenshot of the Turbo C++ IDE window. The title bar reads "Turbo C++ IDE". The main text area shows the output of the program: "Enter two no", "25", "45", "After Swapping value of a is 45", "After Swapping value of b is 25", "Inside Main Function Value of a is 45", and "Inside Main Function Value of b is 25". The output is displayed in a monospaced font on a black background. The window has standard Windows-style controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

34 Program to find largest of two numbers using functions.

ANS.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,big(int,int);
clrscr();
printf("Enter any two no \n");
scanf("%d %d",&a,&b);
c=big(a,b);
printf("Biggest is %d \n",c);
getch();
}
big (int a,int b)
{
if (a>b)
{
return (a);
}
else
return (b);
}
```



35 Program to find factorial of a number using recursion.

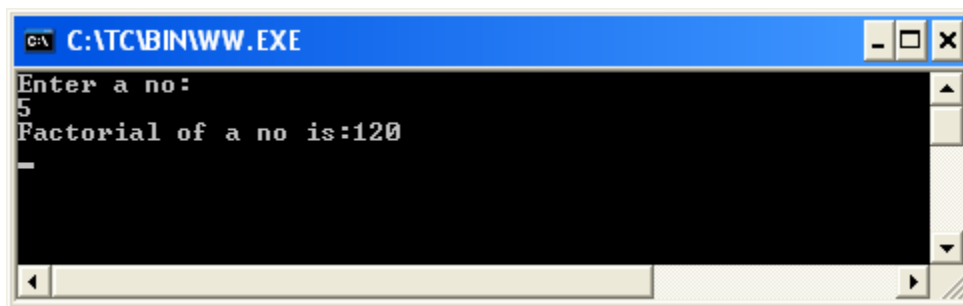
ANS.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,fact(int),f,m;
clrscr();
printf("Enter a no: \n");
```

```

scanf("%d",&n);
f=fact(n);
printf("Factorial of a no is:%d \n",f);
getch();
}
int fact( int m)
{
int f=1;
if(m==1)
return (1);
else
f= m*fact(m-1);
return (f);
}

```



36 Program to display Fibonacci series using recursion

```

#include<stdio.h>

void printFibonacci(int);

void main()
{
    int n;

    printf("Enter the range of the Fibonacci series: ");
    scanf("%d",&n);

    printf("Fibonacci Series: ");
    if(n==1)

```

```

    printf("0");
if(n==2)

    printf("0 1");
else
    Fibonacci(n);

}

void Fibonacci(int n)
{
    int first=0,second=1,sum;

    if(n>0){
        sum = first + second;
        first = second;
        second = sum;
        printf("%d ",sum);
        Fibonacci(n-1);
    }

}

```

Sample output:

Enter the range of the Fibonacci series: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89

```

37 void main()
    {

        int a[10],i,sum=0;
        float av;
        clrscr();
        printf("enter elements of an aarray: ");

        for(i=0;i<10;i++)
            scanf("%d",&a[i]);
        for(i=0;i<10;i++)
            sum=sum+a[i];

        printf("sum=%d",sum);
        av=sum/10;
        printf("average=%.2f",av);
        getch();
    }

```

```
}
```

Output:

enter elements of an array: 4

5

6

1

2

3

5

5

4

7

sum=42

average=4.22

38 Program to find the maximum no in an array.

```
#include<stdio.h>
```

```
int main() {
```

```
    int a[30], i, num, largest;
```

```
    printf("\nEnter no of elements :");
```

```
    scanf("%d", &num);
```

```
    //Read n elements in an array
```

```
    for (i = 0; i < num; i++)
```

```
        scanf("%d", &a[i]);
```

```
    //Consider first element as largest
```

```
    largest = a[0];
```

```
    for (i = 0; i < num; i++) {
```

```
        if (a[i] > largest) {
```

```
            largest = a[i];
```

```
        }
```

```
    }
```

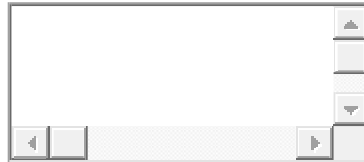
```
    // Print out the Result
```

```
    printf("\nLargest Element : %d", largest);
```

```
return (0);  
}
```

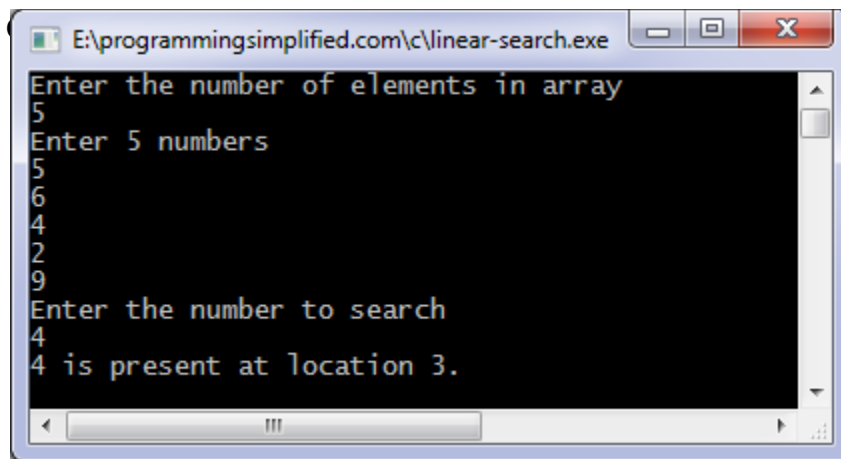
Output :

```
1 Enter no of elements : 5  
2 11 55 33 77 22  
3 Largest Element : 77
```



39 Program to perform Linear search.

```
#include <stdio.h>  
  
int main()  
{  
    int array[100], search, c, n;  
  
    printf("Enter the number of elements in array\n");  
    scanf("%d",&n);  
  
    printf("Enter %d integer(s)\n", n);  
  
    for (c = 0; c < n; c++)  
        scanf("%d", &array[c]);  
  
    printf("Enter the number to search\n");  
    scanf("%d", &search);  
  
    for (c = 0; c < n; c++)  
    {  
        if (array[c] == search)    /* if required element found */  
        {  
            printf("%d is present at location %d.\n", search, c+1);  
            break;  
        }  
    }  
    if (c == n)  
        printf("%d is not present in array.\n", search);  
  
    return 0;  
}
```

program:

40 Program to perform Binary search

```
#include <stdio.h>

int main()
{
    int c, first, last, middle, n, search, array[100];

    printf("Enter number of elements\n");
    scanf("%d",&n);

    printf("Enter %d integers\n", n);

    for (c = 0; c < n; c++)
        scanf("%d",&array[c]);

    printf("Enter value to find\n");
    scanf("%d", &search);

    first = 0;
    last = n - 1;
    middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d found at location %d.\n", search, middle+1);
            break;
        }
        else
```

```

        last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("Not found! %d is not present in the list.\n", search);

    return 0;
}

```

Output

of

program:

```

E:\programmingsimplified.com\c\binary-search.exe
Enter number of elements
7
Enter 7 integers
-4
5
8
9
11
43
485
Enter value to find
11
11 found at location 5.

```

41 Program to perform Bubble sort.

```

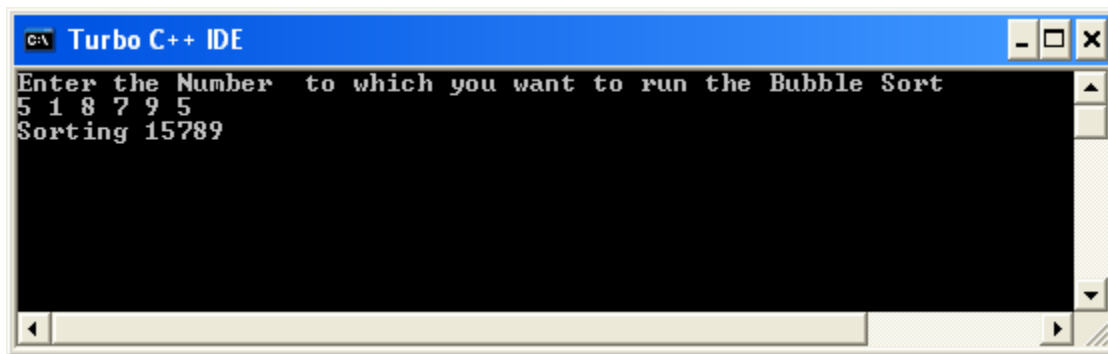
Ans • #include<stdio.h>
#include<conio.h>
void main()
{
    inti,j,n,m;
    int a[50];
    clrscr();
    printf("Enter the Number to which you want to run the Bubble Sort\n");
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for (i=0;i<n-1;i++)
    {
        for (j=0;j<n-i-1;j++)
        {
            if (a[j]>a[j+1])

```

```

{
m=a[j];
a[j]=a[j+1];
a[j+1]=m;
}
}
}
printf("Sorting\t");
for (i=0;i<n;i++)
{
printf("%d",a[i]);
}
printf("\n");
getch();
}

```



42 Program to display matrix.

Ans.

```

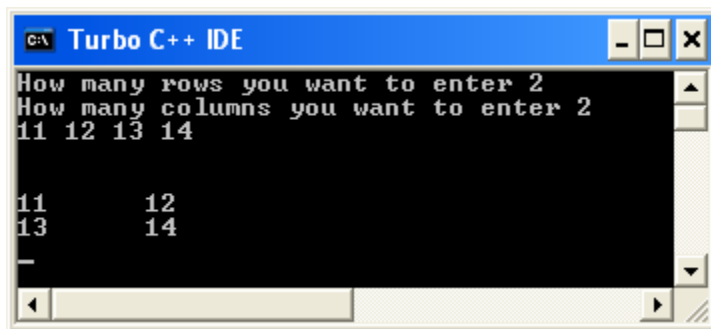
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10];
intr,c,i,j;
clrscr();
printf("How many rows you want to enter");
scanf("%d",&i);
printf("How many columns you want to enter");
scanf("\n%d",&j);
for(r=1;r<=i;r++)
{
for(c=1;c<=j;c++)
{
scanf("%d",&a[r][c]);
}
printf("\n");
}
}

```

```

}
for(r=1;r<=i;r++)
{
for(c=1;c<=j;c++)
{
printf("%d \t",a[r][c]);
}
printf("\n");
}
getch();
}

```



43 Program to find sum of two matrices.

Ans.

```

#include<stdio.h>
#include<conio.h>
void main()
{
inti,j,k,l,m,n,d,e,f,g;
int c[10][10];
int a[10][10],b[10][10];
clrscr();
printf("Enter the Number of Rows and Columns of Matrix A\n");
scanf("%d %d",&k,&l);
printf("Enter the number of Rows and Columns of Matrix B\n");
scanf("%d %d",&m,&n);
printf("Enter the value of Matrix A");
for (i=1;i<=k;i++)
{
for (j=1;j<=l;j++)
{
scanf("%d",&a[i][j]);
}
printf("\n");
}
printf(" Matrix A is: \n");
for (i=1;i<=k;i++)

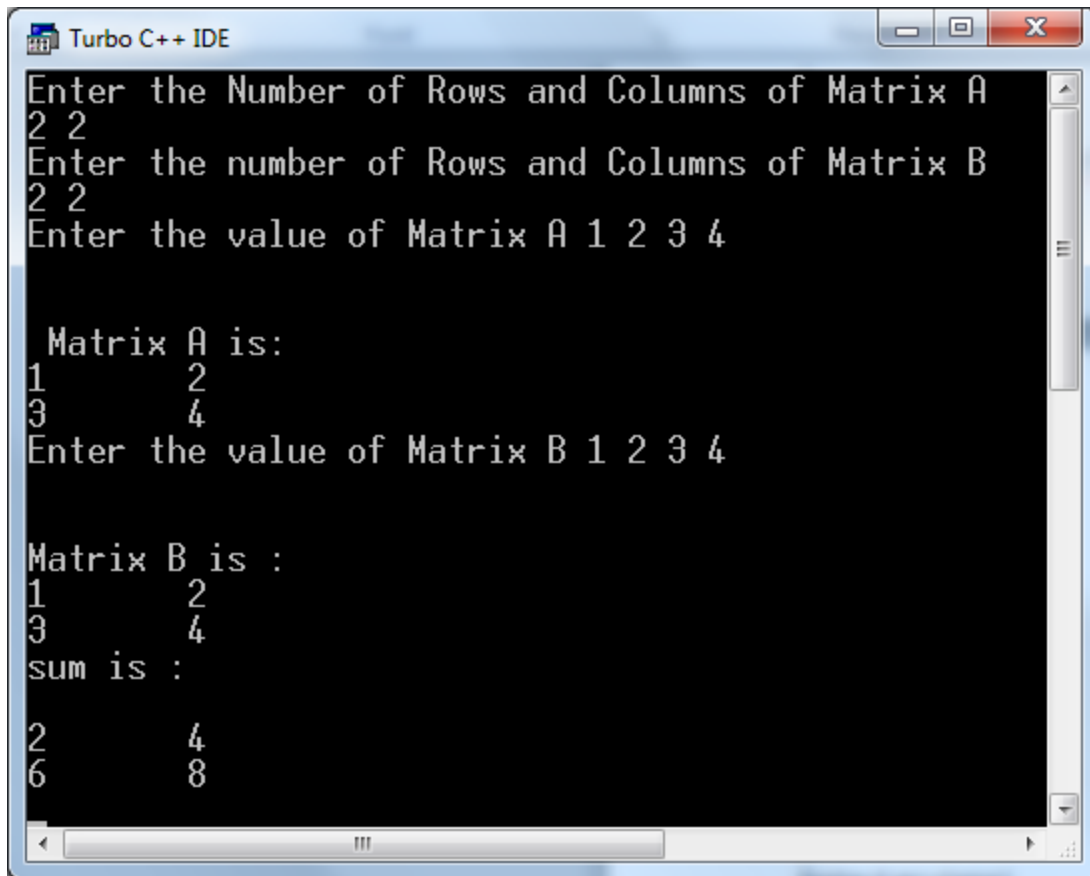
```

```

{
for (j=1;j<=l;j++)
{
printf("%d \t",a[i][j]);
}
printf("\n");
}

printf("Enter the value of Matrix B");
for (d=1;d<=m;d++)
{
for (e=1;e<=n;e++)
{
scanf("%d",&b[d][e]);
}
printf("\n");
}
printf("Matrix B is : \n");
for (d=1;d<=m;d++)
{
for (e=1;e<=n;e++)
{
printf("%d \t",b[d][e]);
}
printf("\n");
}
printf("sum is : \n \n");
for(f=1;f<=k;f++)
{
for(g=1;g<=l;g++)
{
c[f][g]=a[f][g]+b[f][g];
printf("%d\t",c[f][g]);
}
printf("\n");
}
getch();
}

```



```
Turbo C++ IDE
Enter the Number of Rows and Columns of Matrix A
2 2
Enter the number of Rows and Columns of Matrix B
2 2
Enter the value of Matrix A 1 2 3 4

Matrix A is:
1      2
3      4
Enter the value of Matrix B 1 2 3 4

Matrix B is :
1      2
3      4
sum is :

2      4
6      8
```

44 Program to find subtraction of two matrices

```
#include<stdio.h>
#include<conio.h>
void main()
{
    inti,j,k,l,m,n,d,e,f,g;
    int c[10][10];
    int a[10][10],b[10][10];
    clrscr();
    printf("Enter the Number of Rows and Columns of Matrix A\n");
    scanf("%d %d",&k,&l);
    printf("Enter the number of Rows and Columns of Matrix B\n");
    scanf("%d %d",&m,&n);
    printf("Enter the value of Matrix A");
    for (i=1;i<=k;i++)
    {
        for (j=1;j<=l;j++)
        {
            scanf("%d",&a[i][j]);
        }
        printf("\n");
    }
```

```

    }
    printf(" Matrix A is: \n");
    for (i=1;i<=k;i++)
    {
    for (j=1;j<=l;j++)
    {
    printf("%d \t",a[i][j]);
    }
    printf("\n");
    }

    printf("Enter the value of Matrix B");
    for (d=1;d<=m;d++)
    {
    for (e=1;e<=n;e++)
    {
    scanf("%d",&b[d][e]);
    }
    printf("\n");
    }
    printf("Matrix B is : \n");
    for (d=1;d<=m;d++)
    {
    for (e=1;e<=n;e++)
    {
    printf("%d \t",b[d][e]);
    }
    printf("\n");
    }
    printf("Subtraction is : \n \n");
    for(f=1;f<=k;f++)
    {
    for(g=1;g<=l;g++)
    {
    c[f][g]=a[f][g]-b[f][g];
    printf("%d\t",c[f][g]);
    }
    printf("\n");
    }
    getch();
}

```

45 Program to find multiplication of two matrices

```

void main()
{
    int m1[10][10],i,j,k,m2[10][10],add[10][10],mult[10][10];

    printf("enter the first matrix :\n");
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c1;j++)
            scanf("%d\t",&m1[i][j]);
    }
    printf("You have entered the first matrix as follows:\n");

    for(i=0;i<r1;i++)
    {
        for(j=0;j<c1;j++)
            printf("%d\t",m1[i][j]);
        printf("\n");
    }
    for(i=0;i<r2;i++)
    {
        for(j=0;j<c2;j++)
            scanf("%d",&m2[i][j]);
    }
    printf("You have entered the second matrix as follows:\n");
    for(i=0;i<r2;i++)
    {
        for(j=0;j<c2;j++)
            printf("%d\t",m2[i][j]);
        printf("\n");
    }
    printf("The result of the addition is as follows;\n");
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c1;j++)
        {
            add[i][j]=m1[i][j]+m2[i][j];
            printf("%d\t",add[i][j]);
        }
        printf("\n");
    }

    printf("Now we multiply both the above matrix \n");
    printf("The result of the multiplication is as follows:\n");
    /*a11xA11+a12xA21+a13xA31 a11xA12+a12xA22+a13xA32
    a11xA13+a12xA23+a13xA33*/
    for(i=0;i<r1;i++)

```



```

    {
        for(j=0;j<c2;j++)
        {
            mult[i][j]=0;
            for(k=0;k<r1;k++)
            {
                mult[i][j]+=m1[i][k]*m2[k][j];
                /*mult[0][0]=m1[0][0]*m2[0][0]+m1[0][1]*m2[1][0]+m1[0][2]*m2[2][0];*/
            }
            printf("%d\t",mult[i][j]);
        }
        printf("\n");
    }
    getch();
}

```

46 Program to find transpose of a matrix

```

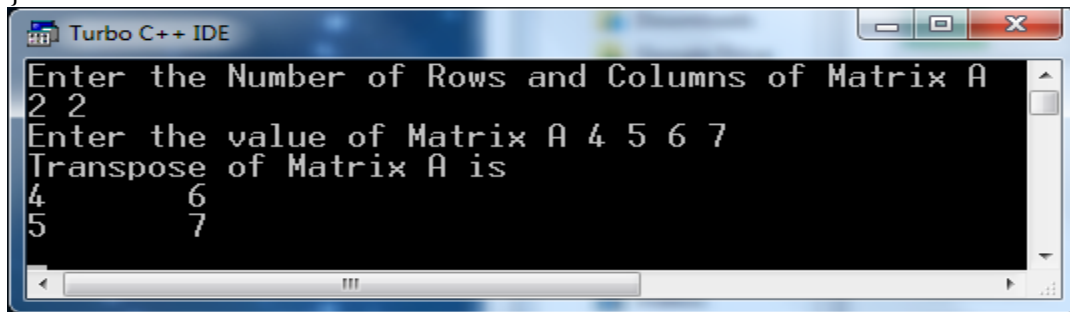
#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,r,c;
    int a[10][10];
    int b[10][10];
    clrscr();
    printf("Enter the Number of Rows and Columns of Matrix A\n");
    scanf("%d %d",&r,&c);
    printf("Enter the value of Matrix A");
    for (i=0;i<r;i++)
    {
        for (j=0;j<c;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for (i=0;i<r;i++)
    {
        for (j=0;j<c;j++)
        {
            b[i][j]=a[j][i];
        }
    }
    printf("Transpose of Matrix A is\n");
    for (i=0;i<r;i++)

```

```

{
for (j=0;j<c;j++)
{
printf("%d \t",b[i][j]);
}
printf("\n");
}
getch();
}

```



47 Program to find the maximum number in array using pointer.

Ans.

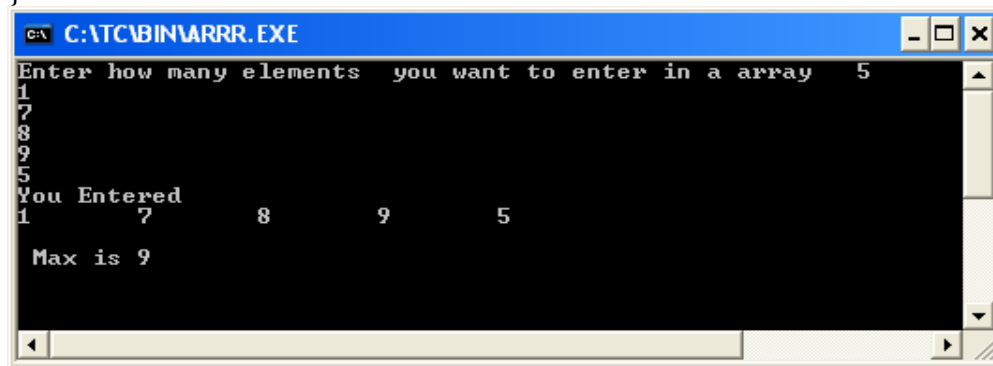
```

#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],i,n,max;
int *p;
clrscr();
printf("Enter how many elements you want to enter in a array \t");
scanf("%d",&n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("You Entered \n");
for (i=0;i<n;i++)
{
printf("%d \t",a[i]);
}
p=&a[0];
max=a[0];
for(i=0;i<n;i++)
{
if(max<=*p)
{
max=*p;
}
}

```

```
p++;  
}
```

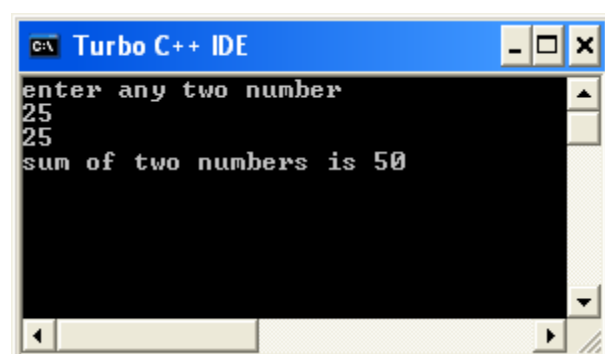
```
printf("\n \n Max is %d",max);  
getch();  
}
```



```
C:\VTC\BIN\ARRR.EXE  
Enter how many elements you want to enter in a array 5  
1  
7  
8  
9  
5  
You Entered  
1 7 8 9 5  
Max is 9
```

48 Program to add two number using pointer.

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int a,b,*pa,*pb,pc;  
clrscr();  
printf("enter any two number\n");  
scanf("%d %d",&a,&b);  
pa=&a;  
pb=&b;  
pc=*pa+*pb;  
printf("sum of two numbers is %d",pc);  
getch();  
}
```

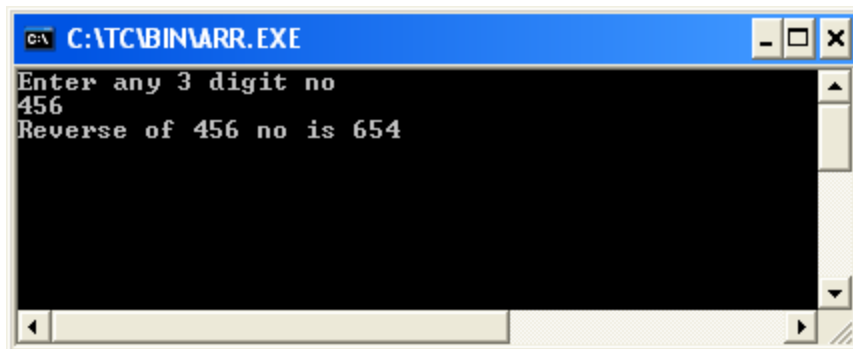


```
Turbo C++ IDE  
enter any two number  
25  
25  
sum of two numbers is 50
```

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int p,n,s=0,*ptr;
    clrscr();
    printf("Enter any 3 digit no \n");
    scanf("%d",&n);
    p=n;
    ptr=&n;
    s=s+(*ptr%10)*100;
    n=*ptr/10;
    s=s+(*ptr%10)*10;
    n=*ptr/10;
    s=s+(*ptr%10)*1;
    n=*ptr/10;
    printf("Reverse of %d no is %d",p,s);
    getch();
}

```



49 Program to find factorial of a number using recursion.

ANS.

```

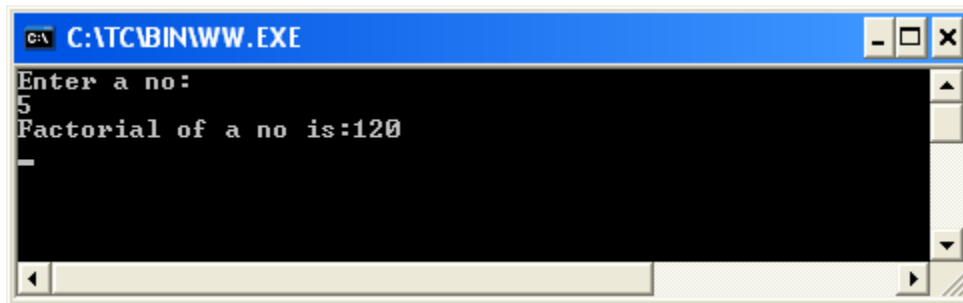
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,fact(int),f,m;
    clrscr();
    printf("Enter a no: \n");
    scanf("%d",&n);
    f=fact(n);
    printf("Factorial of a no is:%d \n",f);
    getch();
}

```

```

int fact( int m)
{
int f=1;
if(m==1)
return (1);
else
f= m*fact(m-1);
return (f);
}

```



50 Program to show input and output of a string.

ANS.

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char name[26];
clrscr();
printf("Enter Your Name:\t");
gets(name);
puts(name);
getch();
}

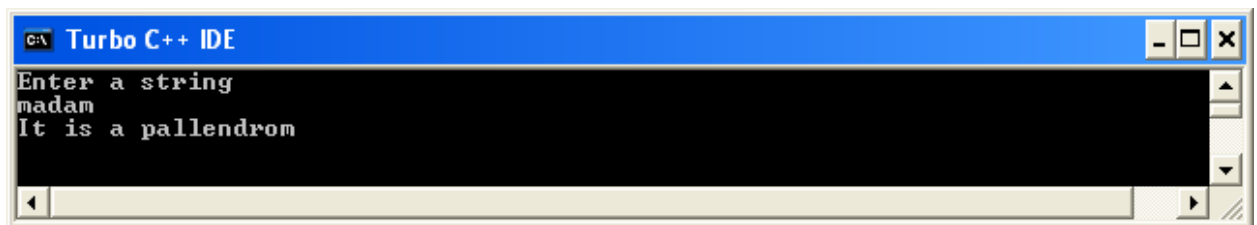
```



51 Program to find whether a string is palindrome or not.

ANS.

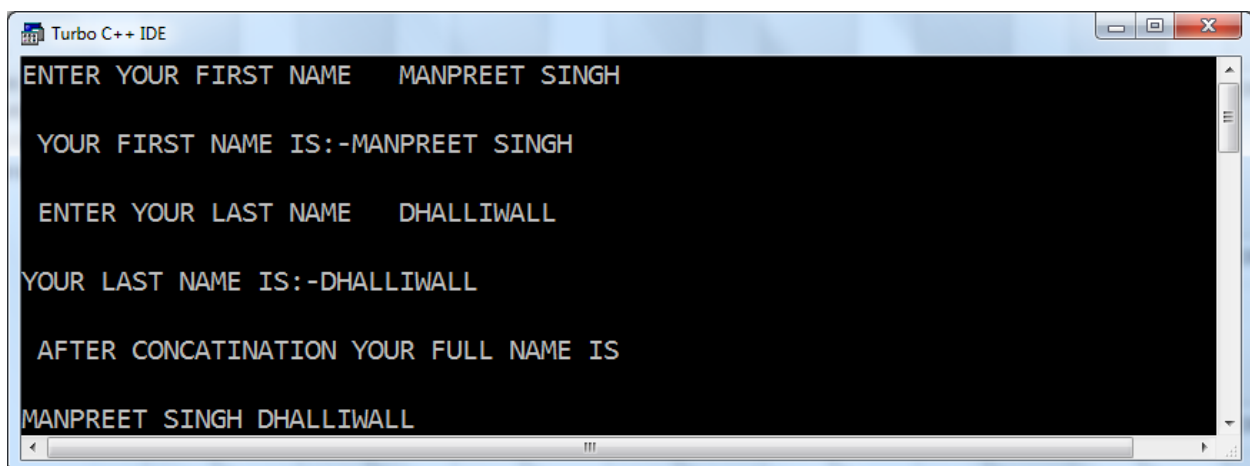
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    inti,l=0,flag;
    charstr[20];
    clrscr();
    printf("Enter a string \n");
    gets(str);
    l=strlen(str);
    for(i=0;i<=l;i++)
    {
        if(str[i]==str[l-1])
        {
            flag=1;
        }
        else
        {
            flag=0;
            break;
        }
        l--;
    }
    if(flag==1)
    {
        printf("It is a pallendrom");
    }
    else
    {
        printf("It is not a pallendrom");
    }
    getch();
}
```



52. WAP to concatenate two strings USING FUNCTION.

ANS.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char a[50],b[50];
clrscr();
printf("ENTER YOUR FIRST NAME \t");
gets(a);
printf("\n YOUR FIRST NAME IS:-%s \t",a);
printf("\n \n ENTER YOUR LAST NAME \t");
gets(b);
printf(" \nYOUR LAST NAME IS:-%s \t",b);
strcat(a," ");
strcat(a,b);
printf("\n \n AFTER CONCATINATION YOUR FULL NAME IS \n\n");
puts(a);
getch();
}
```



```
Turbo C++ IDE
ENTER YOUR FIRST NAME  MANPREET SINGH

YOUR FIRST NAME IS:-MANPREET SINGH

ENTER YOUR LAST NAME  DHALLIWALL

YOUR LAST NAME IS:-DHALLIWALL

AFTER CONCATINATION YOUR FULL NAME IS

MANPREET SINGH DHALLIWALL
```

53 WAP to concatenate two strings WITHOUT USING FUNCTION.

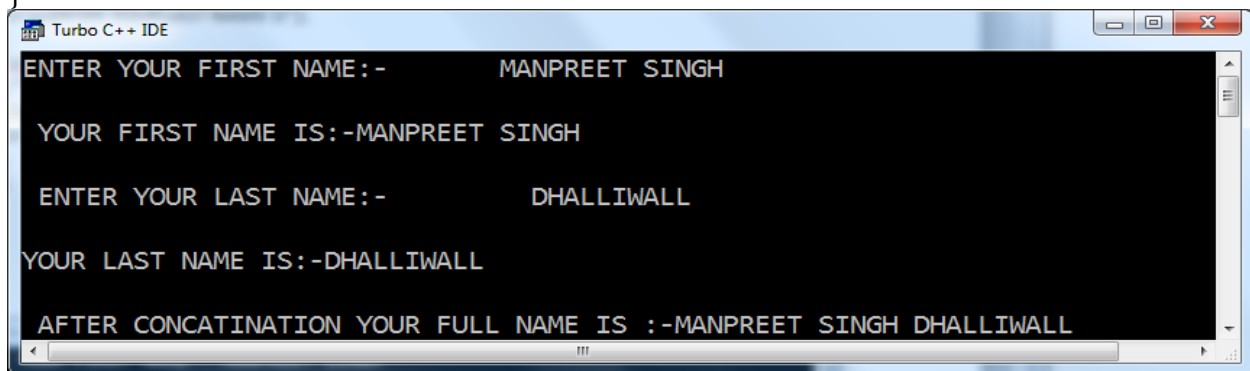
ANS.

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

```

char a[50],b[50];
clrscr();
printf("ENTER YOUR FIRST NAME:- \t");
gets(a);
printf("\n YOUR FIRST NAME IS:-%s \t",a);
printf("\n \n ENTER YOUR LAST NAME:- \t");
gets(b);
printf(" \nYOUR LAST NAME IS:-%s \t",b);
concat(a,b);
printf("\n \n AFTER CONCATINATION YOUR FULL NAME IS :-%s",a);
getch();
}
voidconcat(char a[],char b[])
{
int i=0,j=0;
while(a[i]!='\0')
{
i++;
}
a[i]=' ';
while(b[j]!='\0')
{
a[i+1]=b[j];
j++;
i++;
}
a[i+1]='\0';
}

```



```

Turbo C++ IDE
ENTER YOUR FIRST NAME:-      MANPREET SINGH

YOUR FIRST NAME IS:-MANPREET SINGH

ENTER YOUR LAST NAME:-      DHALLIWALL

YOUR LAST NAME IS:-DHALLIWALL

AFTER CONCATINATION YOUR FULL NAME IS :-MANPREET SINGH DHALLIWALL

```

54. Program to find the length of the string

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
intlen=0,i;
char name[25];

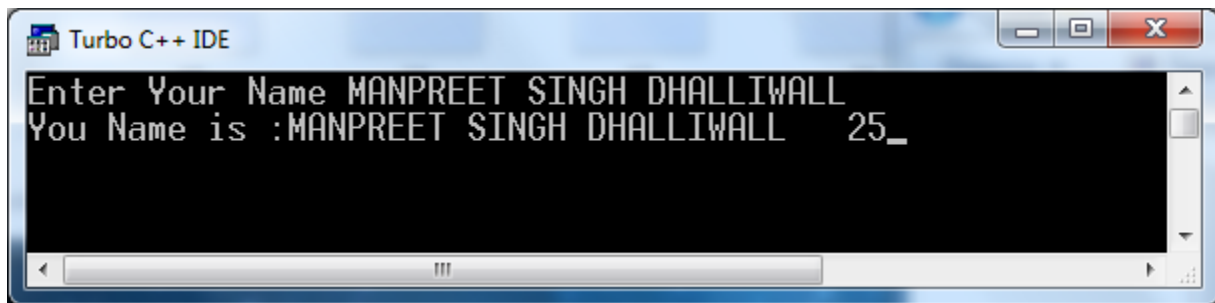
```



```

clrscr();
printf("Enter Your Name\t");
gets(name);
i=0;
while(name[i]!='\0')
{
len=len+1;
i=i+1;
}
printf(" Length Of String is %d",len);
getch();
}

```



55 Program to enter book records

ANS.

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
struct book
{
int id;
char name[30],author[30];
}book[5];
void main()
{
int i;
clrscr();
for(i=0;i<5;i++)
{
printf("Enter Book Id \t Book Name \t Author Name \n");
scanf("\t %d \t %s \t %s",&book[i].id,&book[i].name,&book[i].author);
}
}

```

```

for(i=0;i<5;i++)
{
printf("\n BOOK ID IS =%d \t BOOK NAME IS =%s \t AUTHOR NAME
IS=%s",book[i].id,book[i].name,book[i].author);
}
getch();
}

```

```

C:\TC\BIN\BOOK.EXE
Enter Book Id   Book Name   Author Name
101 C ANKITA_MAM
Enter Book Id   Book Name   Author Name
102 TC NEHA_MAM
Enter Book Id   Book Name   Author Name
103 PHY SK_TIWARI_SIR
Enter Book Id   Book Name   Author Name
104 MATH SHALU_MAM
Enter Book Id   Book Name   Author Name
105 IT KANIKA_MAM

BOOK ID IS =101      BOOK NAME IS =C      AUTHOR NAME IS=ANKITA_MAM
BOOK ID IS =102      BOOK NAME IS =TC      AUTHOR NAME IS=NEHA_MAM
BOOK ID IS =103      BOOK NAME IS =PHY      AUTHOR NAME IS=SK_TIWARI_SIR
BOOK ID IS =104      BOOK NAME IS =MATH      AUTHOR NAME IS=SHALU_MAM
BOOK ID IS =105      BOOK NAME IS =IT      AUTHOR NAME IS=KANIKA_MAM

```

56. Program to enter student record(name, roll no &course).

ANS.

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
struct student
{
char name[50];
int roll;
char course[50];
}student[5];
void main()
{
int i;
clrscr();
for(i=0;i<5;i++)
{
printf("ENTER STUDENT NAME \t ROLLNO \t COURSE \n");
scanf("\t %s\t %d \t %s",&student[i].name,&student[i].roll,&student[i].course);
}
for(i=0;i<5;i++)
{

```

```

printf("\n STUDENT NAME IS  =%s \t ROLL NO IS  =%d \t COURSE IS
=%s",student[i].name,student[i].roll,student[i].course);
}
getch();
}

```

The screenshot shows the Turbo C++ IDE window. The program prompts the user to enter student names, roll numbers, and courses. It then displays the entered data in a formatted table.

STUDENT NAME	IS	ROLL NO	IS	COURSE	IS
RAVINDER	=	71	=	BCA	
BHARAT	=	72	=	BCA	
MOHIT	=	73	=	BCA	
RISHABH	=	74	=	BCA	
BHARAT	=	75	=	BCA	

57 Program to enter student recor(using union)

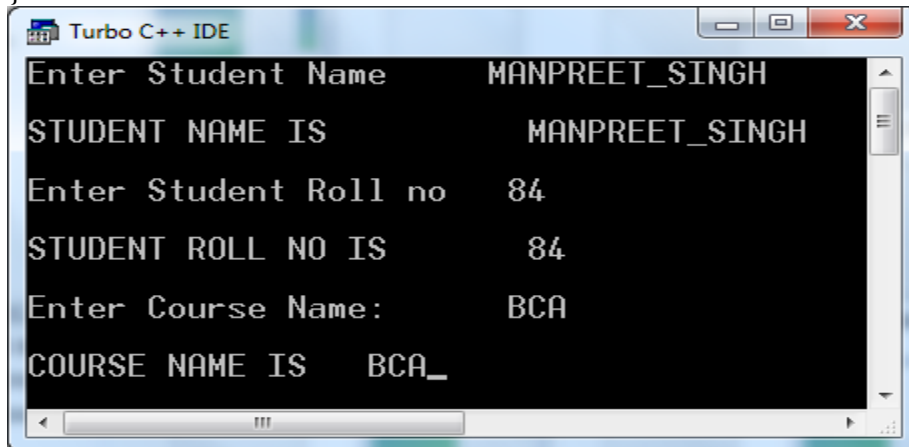
ANS.

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
union student
{
char name[50];
int roll;
char course[50];
}stud;
void main()
{
clrscr();
printf("Enter Student Name \t ");
scanf("%s",&stud.name);
printf("\n");
printf("STUDENT NAME IS \t %s",stud.name);
printf("\n");
printf("\n");
printf("Enter Student Roll no \t");
scanf("%d",&stud.roll);
printf("\n");
printf("STUDENT ROLL NO IS \t %d",stud.roll);
}

```

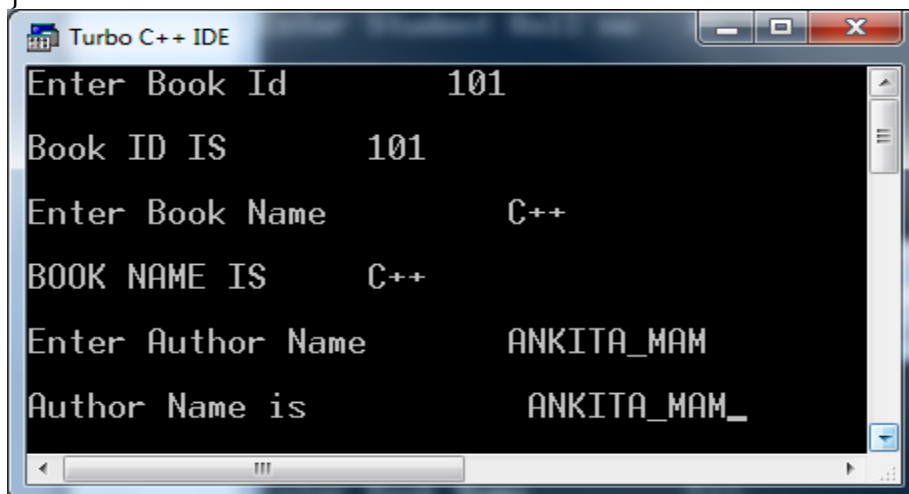
```
printf("\n");  
printf("\n");  
printf("Enter Course Name: \t");  
scanf("%s",&stud.course);  
printf("\n");  
printf("COURSE NAME IS \t %s",stud.course);  
getch();  
}
```



58 Program to enter book records(using union).

ANS.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
union book
{
int id;
char name[30],author[30];
}bo;
void main()
{
clrscr();
printf("Enter Book Id \t");
scanf("%d",&bo.id);
printf("\n");
printf("Book ID IS \t %d",bo.id);
printf("\n");
printf("\n");
printf("Enter Book Name \t");
scanf("%s",&bo.name);
printf("\n");
printf("BOOK NAME IS \t %s",bo.name);
printf("\n");
printf("\n");
printf("Enter Author Name \t");
scanf("%s",&bo.author);
printf("\n");
printf("Author Name is \t %s",bo.author);
getch();
}
```



59 WRITE A PROGRAM TO CREATE A FILE & READ THE CONTENT IN THE FILE.

ANS.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
FILE *fp;
char c='.';
clrscr();
fp=fopen("FILE.txt","w");
if(fp==NULL)
{
printf("File Does Not Exit \n");
exit(1);
}
printf("Write the Statement and put '.' to stop \n");
do
{
c=getche();
fputc(c,fp);
}
while(c!='.');
fclose(fp);
printf("\n Start Reading \n");
fp=fopen("FILE.txt","r");
do
{
c=getc(fp);
putchar (c);
}while (c!=EOF);
getch();
}
```



60. WAP to copy the content of one file into another file.

ANS.

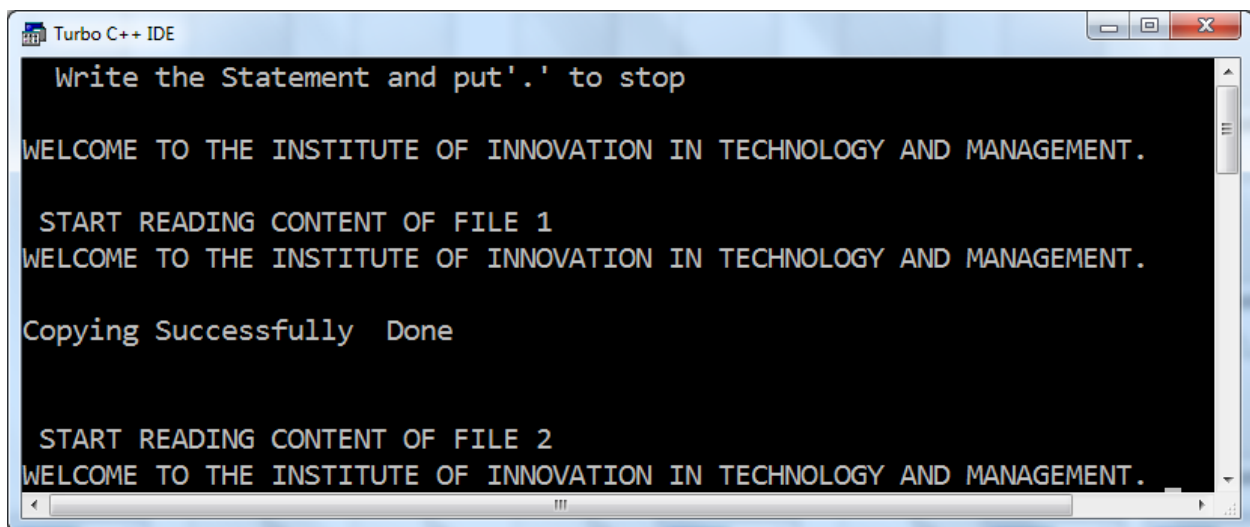
```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
FILE *fp1,*fp2;
char c='.';
clrscr();
fp1=fopen("FILE 1.txt","w");
if(fp1==NULL||fp2==NULL)
{
printf("File Does Not Exit \n");
exit(1);
}
printf("\t Write the Statement and put '.' to stop \n \n");
do
{
c=getche();
fputc(c,fp1);
}
while(c!='.');
fclose(fp1);
printf("\n\n START READING CONTENT OF FILE 1\n");
fp1=fopen("FILE 1.txt","r");
do
{
c=fgetc(fp1);
putchar(c);
}while (c!=EOF);
fclose(fp1);
printf("\n\nCopying Successfully Done \n");
fp1=fopen("FILE 1.txt","r");
fp2=fopen("FILE 2.txt","w");
do
{
c=fgetc(fp1);
fputc(c,fp2);
}while(c!=EOF);
fclose(fp1);
fclose(fp2);

printf("\n\n START READING CONTENT OF FILE 2\n");
fp2=fopen("FILE 2.txt","r");
do
```

```

{
c=fgetc(fp2);
putchar(c);
}while (c!=EOF);
fclose(fp2);
getch();
}

```



```

Turbo C++ IDE
Write the Statement and put\'.\' to stop

WELCOME TO THE INSTITUTE OF INNOVATION IN TECHNOLOGY AND MANAGEMENT.

START READING CONTENT OF FILE 1
WELCOME TO THE INSTITUTE OF INNOVATION IN TECHNOLOGY AND MANAGEMENT.

Copying Successfully Done

START READING CONTENT OF FILE 2
WELCOME TO THE INSTITUTE OF INNOVATION IN TECHNOLOGY AND MANAGEMENT.

```

61 Program to show the use of enum data type

```
#include <stdio.h>
```

```

int main ()
{
enum planets
{
Mercury,
Venus,
Earth,
Mars,
Jupiter,
Saturn,
Uranus,
Neptune,
Pluto
};

```



```

enum planets planet1, planet2;

planet1 = Mars;
planet2 = Earth;

if (planet1 > planet2)
    puts ("Mars is farther from the Sun than Earth is.");
else
    puts ("Earth is farther from the Sun than Mars is.");

return 0;
}

```

Output

Mars is farther from the Sun than Earth is.

62 Program to count number of lines, words and characters in a file.

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int noc=0,now=0,nol=0;
    FILE *fw,*fr;
    char fname[20],ch;
    clrscr();
    printf("\n enter the source file name");
    gets(fname);
    fr=fopen(fname,"r");
    if(fr==NULL)
    {
        printf("\n error \n");
        exit(0);
    }
    ch=fgetc(fr);
    while(ch!=EOF)
    {
        noc++;
        if(ch==' ');
        now++;
        if(ch=='\n')
        {
            nol++;
            now++;
        }
    }
}

```

```

}
ch=fgetc(fr);
}
fclose(fr);
printf("\n total no of character=%d",noc);
printf("\n total no of words=%d",now);
printf("\n total no of lines=%d",nol);
getch();
}

```

63 Program to copy the content of one file into another file using command line arguments.

```

#include<stdio.h>
#include<conio.h>
main(int argc,char *argv[])
{
FILE *fp1,*fp2;
char ch;
clrscr();
if(argc!=3)
{
printf("\n insufficient argument ");
exit(0);
}
fp1=fopen(argv[1],"r");
fp2=fopen(argv[2],"w");
if(fp1==NULL || fp2==NULL)
{
printf("\n unable to open file ");
exit(0);
}
while(!feof(fp1))
{
ch=fgetc(fp1);
fputc(ch,fp2);
}
printf("\n file successfully copied ");
fclose(fp1);
fclose(fp2);
getch();
}

```

ADVANCE PROBLEMS

1. Write a program to transfer each line of first file to second file

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp1,*fp2;
char ch,fname1[20],fname2[20];
printf("\n enter source file name");
gets(fname1);
printf("\n enter sourse file name");
gets(fname2);
fp1=fopen(fname1,"r");
fp2=fopen(fname2,"w");
if(fp1==NULL||fp2==NULL)
{
printf("unable to open");
exit(0);
}
do
{
ch=fgetc(fp1);
fputc(ch,fp2);
}
while(ch!=EOF);
fcloseall();
getch();
}
```

2. Write a program to count the number of vowels in a given string

Logic: Here variable 'vowels' is incremented whenever a vowel found in tracing. Logic behind this is very simple, that comparing each character to the set of vowels, predefined in an array. If

this character is in the set of vowels then it implies that character is a vowel, so we increment the count by one.

Same logic can be applied to find the existence of any letter set in the predetermined array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char str[50];
int vowels = 0, i = 0;
clrscr();
printf("\n\n\t ENTER A STRING...: ");
gets(str);
while(str[i] != '\0')
{
    if(str[i]=='A' || str[i]=='a' || str[i]=='E' || str[i]=='e' || str[i]=='I' || str[i]=='i' ||
       str[i]=='O' || str[i]=='o' || str[i]=='U' || str[i]=='u')
    {
        vowels++;
    }

    i++;
}
printf("\n\n\t THE TOTAL NUMBER OF VOWELS IS...: %d", vowels);
getch();
}
```

3. Armstrong number c program:

c programming code to check whether a number is armstrong or not. A number is armstrong if the sum of cubes of individual digits of a number is equal to the number itself. For example 371 is an armstrong number as $3^3 + 7^3 + 1^3 = 371$. Some other armstrong numbers are: 0, 1, 153, 370, 407.

C programming code

```
#include <stdio.h>

int main()
{
    int number, sum = 0, temp, remainder;
```

```

printf("Enter an integer\n");
scanf("%d",&number);

temp = number;

while( temp != 0 )
{
    remainder = temp%10;
    sum = sum + remainder*remainder*remainder;
    temp = temp/10;
}

if ( number == sum )
    printf("Entered number is an armstrong number.\n");
else
    printf("Entered number is not an armstrong number.\n");

return 0;
}

```

4. Pointer to a pointer

A pointer to a pointer is a form of multiple indirection, or a chain of pointers. Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.



```
#include <stdio.h>
```

```
void main ()
```

```
{
```

```
    int var;
```

```
    int *ptr;
```

```
    int **pptr;
```

```
    var = 3000;
```

```
    /* take the address of var */

```

```

ptr = &var;

/* take the address of ptr using address of operator & */
pptr = &ptr;

/* take the value using pptr */
printf("Value of var = %d\n", var );
printf("Value available at *ptr = %d\n", *ptr );
printf("Value available at **pptr = %d\n", **pptr);

getch();
}

```

When the above code is compiled and executed, it produces the following result:

```

Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000

```

5. self referential structure

```

#include <stdio.h>
struct stud
{
int roll;
char name[30];
int age;
struct stud *next;
}
main()
{
struct stud n1, n2, n3;
struct stud *p;
scanf ("%d %s %d", &n1.roll, n1.name, &n1.age);
scanf ("%d %s %d", &n2.roll, n2.name, &n2.age);
scanf ("%d %s %d", &n3.roll, n3.name, &n3.age);
n1.next = &n2;
n2.next = &n3;
n3.next = NULL;
/* Now traverse the list and print the elements */
p = &n1; /* point to 1st element */
while (p != NULL)
{
printf ("\n %d %s %d", p->roll, p->name, p->age);

```

```
p = p->next;  
}  
}
```

6. program to read values from file and store even values and odd values in separate files

```
#include<stdio.h>  
#include<conio.h>  
  
void main()  
{  
    FILE *f1,*f2,*f3;  
    int number,i;  
    clrscr();  
  
    printf("Contents of DATA file\n\n");  
    f1 = fopen("bcaDATA","w"); /* create a data file */  
    for(i=1;i<=5;i++)  
    {  
        scanf("%d",&number);  
        if(number==-1)break;  
        putw(number,f1);  
    }  
    fclose(f1);  
  
    f1 = fopen("bcaDATA","r");  
    f2 = fopen("bcaODD","w");  
    f3 = fopen("bcaEVEN","w");  
  
    while((number = getw(f1)) != EOF) /* Read from Data file */  
    {  
        if(number%2==0)  
            putw(number,f3);  
        else  
            putw(number,f2);  
    }  
    fclose(f1);  
    fclose(f2);  
    fclose(f3);  
  
    f2 = fopen("bcaODD","r");  
    f3 = fopen("bcaEVEN","r");
```

```
printf("\n\nContents of ODD file \n\n");
while((number = getw(f2)) != EOF)
    printf("%4d",number);

printf("\n\nContents of EVEN file \n\n");
while((number = getw(f3)) != EOF)
    printf("%4d",number);

fclose(f2);
fclose(f3);
getch();
}
```


Viva Questions

1. What is the difference between declaring a variable and defining a variable?

Declaration of a variable in C hints the compiler about the type and size of the variable in compile time. Similarly, declaration of a function hints about type and size of function parameters. No space is reserved in memory for any variable in case of declaration.

Example: `int a;`

Here variable 'a' is declared of data type 'int'

Defining a variable means declaring it and also allocating space to hold it.

We can say "Definition = Declaration + Space reservation".

Example: `int a = 10;`

Here variable "a" is described as an int to the compiler and memory is allocated to hold value 10.

2. What is a static variable?

A static variable is a special variable that is stored in the data segment unlike the default automatic variable that is stored in stack. A static variable can be initialized by using keyword static before variable name.

Example:

`static int a = 5;`

A static variable behaves in a different manner depending upon whether it is a global variable or a local variable.

A static global variable is same as an ordinary global variable except that it cannot be accessed by other files in the same program / project even with the use of keyword extern. A static local variable is different from local variable. It is initialized only once no matter how many times that function in which it resides is called. It may be used as a count variable.

Example:

```
#include <stdio.h>
```

```
//program in file f1.c
```

```
void count(void) {
```

```
static int count1 = 0;
```

```
int count2 = 0;
```

```
count1++;
```

```
count2++;
```

```
printf("\nValue of count1 is %d, Value of count2 is %d", count1, count2);
```

```
}/
```

```
*Main function*/
```

```
int main(){
```

```
count();
```

```
count();
```

```
count();  
return 0;  
}
```

Output:

Value of count1 is 1, Value of count2 is 1

Value of count1 is 2, Value of count2 is 1

Material from Interview Mantra. Subscribe to free updates via email.

Value of count1 is 3, Value of count2 is 1

3. What is a register variable?

Register variables are stored in the CPU registers. Its default value is a garbage value. Scope of a register variable is local to the block in which it is defined. Lifetime is till control remains within the block in which the register variable is defined. Variable stored in a CPU register can always be accessed faster than the one that is stored in memory. Therefore, if a variable is used at many places in a program, it is better to declare its storage class as register

Example:

```
register int x=5;
```

Variables for loop counters can be declared as register. Note that register keyword may be ignored by some compilers.

4. Where is an auto variables stored?

Main memory and CPU registers are the two memory locations where auto variables are stored. Auto variables are defined under automatic storage class. They are stored in main memory. Memory is allocated to an automatic variable when the block which contains it is called and it is de-allocated at the completion of its block execution.

Auto variables:

Storage : main memory.

Default value : garbage value.

Scope : local to the block in which the variable is defined.

Lifetime : till the control remains within the block in which the variable is defined.

5. What is scope & storage allocation of extern and global variables?

Extern variables: belong to the External storage class and are stored in the main memory. extern is used when we have to refer a function or variable that is implemented in other file in the same project. The scope of the extern variables is Global.

Example:

```
/******
```

Index: f1.c

```
*****
```

```
#include <stdio.h>
```

```
extern int x;
```

```
int main() {
```

```
printf("value of x %d", x);
return 0;
}
```

Index: f2.c

```
*****/
```

```
int x = 3;
```

Here, the program written in file f1.c has the main function and reference to variable x. The file f2.c has the declaration of variable x. The compiler should know the datatype of x and this is done by *extern* definition.

Material from Interview Mantra. Subscribe to free updates via email.

Global variables: are variables which are declared above the main() function. These variables are accessible throughout the program. They can be accessed by all the functions in the program. Their default value is zero.

Example:

```
#include <stdio.h>
```

```
int x = 0;
```

```
/* Variable x is a global variable.
```

```
It can be accessed throughout the program */
```

```
void increment(void) {
```

```
x = x + 1;
```

```
printf("\n value of x: %d", x);
```

```
} int main(){
```

```
printf("\n value of x: %d", x);
```

```
increment();
```

```
return 0;
```

```
}
```

6. What is scope & storage allocation of register, static and local variables?

Register variables: belong to the register storage class and are stored in the CPU registers. The scope of the register variables is local to the block in which the variables are defined. The variables which are used for more number of times in a program are declared as register variables for faster access. Example: loop counter variables.

```
register int y=6;
```

Static variables: Memory is allocated at the beginning of the program execution and it is reallocated only after the program terminates. The scope of the static variables is local to the block in which the variables are defined.

Example:

```
#include <stdio.h>
```

```
void decrement(){
```

```
static int a=5;
```

```
a--;
```

```
printf("Value of a:%d\n", a);
```

```
} int main(){
```

```
decrement();
```

```
return 0;
```

}

Here 'a' is initialized only once. Every time this function is called, 'a' does not get initialized. so output would be 4

3 2 etc.,

Local variables: are variables which are declared within any function or a block. They can be accessed only by function or block in which they are declared. Their default value is a garbage value.

7. What are storage memory, default value, scope and life of Automatic and Register storage class?

1. Automatic storage class:

Storage : main memory.

Default value : garbage value.

Material from Interview Mantra. Subscribe to free updates via email.

Scope : local to the block in which the variable is defined.

Lifetime : till control remains within the block.

2. Register storage class:

Storage : CPU registers.

Default value : garbage value.

Scope : local to the block in which the variable is defined.

Lifetime : till control remains within the block.

8. What are storage memory, default value, scope and life of Static and External storage class?

1. Static storage class:

Storage : main memory.

Default value : zero

Scope : local to the block in which the variable is defined.

Lifetime : till the value of the variable persists between different function calls.

2. External storage class:

Storage : main memory

Default value : zero

Scope : global

Lifetime : as long as the program execution doesn't come to an end.

9. What is the difference between 'break' and 'continue' statements?

Differences between 'break' and 'continue' statements

break continue

1. break is a keyword used to terminate the loop or exit from the block. The control jumps to next statement after the loop or block.

1. continue is a keyword used for skipping the current iteration and go to next iteration of the loop

2.Syntax:

```

{
Statement 1;
Statement 2;
Statement n;
break;
}

```

2. Syntax:

```

{
Statement 1;
continue;
Statement 2;
}

```

3. break can be used with for, while, do- while, and switch statements. When break is used in nested loops i.e. within the inner most loop then only the innermost loop is terminated.

3. This statement when occurs in a loop does not terminate it but skips the statements after this continue statement. The control goes to the next iteration. Continue can be used with for, while and do-while.

4. Example:

```

i = 1, j = 0;
while(i<=5)

```

```

{
i=i+1;
if(i== 2)

```

4. Example:

```

i = 1, j = 0;
while(i<=5)

```

```

{
i=i+1;
if(i== 2)

```

```

break;

```

```

j=j+1;

```

```

}

```

```

continue;

```

```

j=j+1;

```

```

}

```

10. What is the difference between 'for' and 'while' loops?

for loop: When it is desired to do initialization, condition check and increment/decrement in a single statement of an iterative loop, it is recommended to use 'for' loop.

Syntax:

```

for(initialization;condition;increment/decrement)

```

```

{

```

```

/block of statements

```

```

increment or decrement

```

```
}
```

Program: Program to illustrate for loop

```
#include<stdio.h>
```

```
int main() {
```

```
int i;
```

```
for (i = 1; i <= 5; i++) {
```

```
//print the number
```

```
printf("\n %d", i);
```

```
}
```

```
return 0;
```

```
}
```

Output:

12345

Explanation:

The loop repeats for 5 times and prints value of 'i' each time. 'i' increases by 1 for every cycle of loop.

while loop: When it is not necessary to do initialization, condition check and increment/decrement in a single

statement of an iterative loop, while loop could be used. In while loop statement, only condition statement is

present.

Syntax:

```
#include<stdio.h>
```

```
int main() {
```

```
int i = 0, flag = 0;
```

```
int a[10] = { 0, 1, 4, 6, 89, 54, 78, 25, 635, 500 };
```

```
//This loop is repeated until the condition is false.
```

Material from Interview Mantra. Subscribe to free updates via email.

```
while (flag == 0) {
```

```
if (a[i] == 54) {
```

```
//as element is found, flag = 1,the loop terminates
```

```
flag = 1;
```

```
}
```

```
else {
```

```
i++;
```

```
}
```

```
}
```

```
printf("Element found at %d th location", i);
```

```
return 0;
```

```
}
```

Output:

Element found at 5th location

Explanation:

Here flag is initialized to zero. 'while' loop repeats until the value of flag is zero, increments i by 1. 'if' condition

checks whether number 54 is found. If found, value of flag is set to 1 and 'while' loop

terminates.

Which bitwise operator is suitable for checking whether a particular bit is ON or OFF?

Bitwise AND operator.

Example: Suppose in byte that has a value 10101101 . We wish to check whether bit number 3 is ON (1) or OFF

(0) . Since we want to check the bit number 3, the second operand for AND operation we choose is binary

00001000, which is equal to 8 in decimal.

Explanation:

ANDing operation :

10101101 original bit pattern

00001000 AND mask

00001000 resulting bit pattern

The resulting value we get in this case is 8, i.e. the value of the second operand. The result turned out to be a 8 since the third bit of operand was ON. Had it been OFF, the bit number 3 in the resulting bit pattern would have evaluated to 0 and complete bit pattern would have been 00000000. Thus depending upon the bit number to be checked in the first operand we decide the second operand, and on ANDing these two operands the result decides whether the bit was ON or OFF.

2. Which bitwise operator is suitable for turning OFF a particular bit in a number?

Bitwise AND operator (&), one's complement operator(~)

Example: To unset the 4th bit of byte_data or to turn off a particular bit in a number.

Explanation:

Consider,

Material from Interview Mantra. Subscribe to free updates via email.

char byte_data= 0b00010111;

byte_data= (byte_data)&(~(1<<4));

1 can be represented in binary as 0b00000001 = (1<<4)

<< is a left bit shift operator,

it shifts the bit 1 by 4 places towards left.

(1<<4) becomes 0b00010000

And ~ is the one's complement operator in C language.

So ~(1<<4) = complement of 0b00010000

= 0b11101111

Replacing value of byte_data and ~(1<<4) in

(byte_data)&(~(1<<4));

we get (0b00010111) & (0b11101111)

Perform AND operation to below bytes.

00010111

11101111

00000111

Thus the 4th bit is unset.

3. What is equivalent of multiplying an unsigned int by 2: left shift of number by 1 or right shift of number by 1?

Left shifting of an unsigned integer is equivalent to multiplying an unsigned int by 2.

Eg1: $14 \ll 1$;

Consider a number 14-----00001110 (8+4+2) is its binary equivalent
left shift it by 1-----00011100 (16+8+4) which is 28.

Eg2: $1 \ll 1$;

consider the number as 1---00000001 (0+0+1).

left shift that by 1-----00000010 (0+2+0) which is 2.

left shift by 1 bit of a number = $2 * \text{number}$

left shift by 1 bit of $2 * \text{number} = 2 * 2 * \text{number}$

left shift by n bits of number = $(2^n) * \text{number}$

Program: Program to illustrate left shift and right shift operations.

```
#include<stdio.h>
```

```
int main(void)
```

```
{
```

```
int x=10,y=10;
```

```
printf("left shift of 10 is %d \n",x<<1);
```

```
printf("right shift of 10 is %d \n",y>>1);
```

```
return 0;
```

```
}
```

Output:

left shift of 10 is 20

right shift of 10 is 5

Explanation:

Left shift (by 1 position) multiplies a number by two. Right shift divides a number by 2.

Material from Interview Mantra. Subscribe to free updates via email.

4. What is an Enumeration Constant?

Enumeration is a data type. We can create our own data type and define values that the variable can take. This can help in making program more readable. enum definition is similar to that of a structure.

Example: consider light_status as a data type. It can have two possible values - on or off.

```
enum light_status
```

```
{
```

```
on, off
```

```
};
```

```
enum light_status bulb1, bulb2;
```

```
/* bulb1, bulb2 are the variables */
```

Declaration of enum has two parts:

a) First part declares the data type and specifies the possible values, called 'enumerators'.

b) Second part declares the variables of this data type.

We can give values to these variables:

```
bulb1 = on;
```



```
bulb2 = off;
```

5. What is a structure?

A structure is a collection of pre-defined data types to create a user-defined data type. Let us say we need to create records of students. Each student has three fields:

```
int roll_number;  
char name[30];  
int total_marks;
```

This concept would be particularly useful in grouping data types. You could declare a structure student as:

```
struct student {  
int roll_number;  
char name[30];  
int total_marks;  
} student1, student2;
```

The above snippet of code would declare a structure by name student and it initializes two objects student1,

student2. Now these objects and their fields could be accessed by saying student1.roll_number for accessing roll number field of student1 object, similarly student2.name for accessing name field of student2 object.

6. What are the differences between a structure and a union?

Structures and Unions are used to store members of different data types.

STRUCTURE UNION

a)Declaration:

```
struct  
{  
data type member1;  
data type member2;  
};
```

a)Declaration:

```
union  
{  
data type member1;  
data type member2;  
};
```

b)Every structure member is allocated memory when a structure variable is defined.

Example:

b)The memory equivalent to the largest item is allocated commonly for all members.

Example:

Material from Interview Mantra. Subscribe to free updates via email.

```
struct emp {  
char name[5];  
int age;  
float sal;  
};  
struct emp e1;
```

Memory allocated for structure is $1+2+4=7$ bytes. 1

byte for name, 2 bytes for age and 4 bytes for sal.

```
union emp1 {  
char name[5];  
int age;  
float sal;  
};  
union emp1 e2;
```

Memory allocated to a union is equal to size of the largest member. In this case, float is the largest-sized data type. Hence memory allocated to this union is 4 bytes.

c)All structure variables can be initialized at a time

```
struct st {  
int a;  
float b;  
};  
struct st s = { .a=4, .b=10.5 };
```

Structure is used when all members are to be independently used in a program.

c)Only one union member can be initialized at a time

```
union un {  
int a;  
float b;  
};  
union un un1 = { .a=10 };
```

Union is used when members of it are not required to be accessed at the same time.

7. What are the advantages of unions?

Union is a collection of data items of different data types. It can hold data of only one member at a time though it has members of different data types. If a union has two members of different data types, they are allocated the same memory. The memory allocated is equal to maximum size of the members. The data is interpreted in bytes depending on which member is being accessed.

Example:

```
union pen {  
char name;  
float point;  
};
```

Here name and point are union members. Out of these two variables, 'point' is larger variable which is of float data type and it would need 4 bytes of memory. Therefore 4 bytes space is allocated for both the variables. Both the variables have the same memory location. They are accessed according to their type. Union is efficient when members of it are not required to be accessed at the same time.

8. How can typedef be to define a type of structure?

typedef declaration helps to make source code of a C program more readable. Its purpose is to redefine the name of an existing variable type. It provides a short and meaningful way to call a data type. typedef is useful when the name of the data type is long. Use of typedef can reduce length and complexity of data types. Note: Usually uppercase letters are used to make it clear that we are dealing with our own data type.

Example:

```
struct employee {  
char name[20];  
int age;  
};
```

Material from Interview Mantra. Subscribe to free updates via email.

```
struct employee e;
```

The above declaration of the structure would be easy to use when renamed using typedef as:

```
struct employee {  
char name[20];  
int age;  
};  
typedef struct employee EMP;  
EMP e1, e2;
```

9. Write a program that returns 3 numbers from a function using a structure.

A function in C can return only one value. If we want the function to return multiple values, we need to create a structure variable, which has three integer members and return this structure.

Program: Program with a function to return 3 values

```
#include<stdio.h>  
//sample structure which has three integer variables.  
struct sample {  
int a, b, c;  
};  
//this is function which returns three values.  
struct sample return3val() {  
struct sample s1;  
s1.a = 10;  
s1.b = 20;  
s1.c = 30;  
//return structure s1, which means return s1.a ,s1.b and s1.c  
return s1;  
}  
int main() {  
struct sample accept3val;  
//three values returned are accepted by structure accept3val.  
accept3val = return3val();  
//prints the values  
printf(" \n %d", accept3val.a);  
printf("\n %d", accept3val.b);  
printf(" \n %d", accept3val.c);  
return 0;  
}
```

Output:

10
20
30.

Explanation:

In this program, we use C structure to return multiple values from a function. Here we have a structure holding three int variables and a function which returns it. 'return3val' is a function which assigns 10, 20, 30 to its integer variables and returns this structure. In this program, 'accept3val' is a structure used to accept the values returned by the function. It accepts those values and shows the output. Material from Interview Mantra. Subscribe to free updates via email.

CHAPTER : Functions

1. What is the purpose of main() function?

In C, program execution starts from the main() function. Every C program must contain a main() function. The main function may contain any number of statements. These statements are executed sequentially in the order which they are written.

The main function can in-turn call other functions. When main calls a function, it passes the execution control to that function. The function returns control to main when a return statement is executed or when end of function is reached.

In C, the function prototype of the 'main' is one of the following:

```
int main(); //main with no arguments
```

```
int main(int argc, char *argv[]); //main with arguments
```

The parameters argc and argv respectively give the number and value of the program's command-line arguments.

Example:

```
#include <stdio.h>
/* program section begins here */
int main() {
// opening brace - program execution starts here
printf("Welcome to the world of C");
return 0;
}/
/ closing brace - program terminates here
```

Output:

Welcome to the world of C

2. Explain command line arguments of main function?

In C, we can supply arguments to 'main' function. The arguments that we pass to main () at command prompt are called command line arguments. These arguments are supplied at the time of invoking the program.

The main () function can take arguments as: main(int argc, char *argv[]) { }

The first argument argc is known as 'argument counter'. It represents the number of arguments in the command line. The second argument argv is known as 'argument vector'. It is an array of char type pointers that points to the command line arguments. Size of this array will be equal to the value of argc. **Example:** at the command prompt if

we give:

C:\> fruit.exe apple mango

then

argc would contain value 3

argv [0] would contain base address of string " fruit.exe" which is the command name that invokes the program.

argv [1] would contain base address of string "apple"

argv [2] would contain base address of string "mango"

here apple and mango are the arguments passed to the program fruit.exe

Program:

```
#include <stdio.h>
```

```
int main(int argc, char *argv[]) {
```

Material from Interview Mantra. Subscribe to free updates via email.

```
int n;
```

```
printf("Following are the arguments entered in the command line");
```

```
for (n = 0; n < argc; n++) {
```

```
printf("\n %s", argv[n]);
```

```
}
```

```
printf("\n Number of arguments entered are\n %d\n", argc);
```

```
return 0;
```

```
}
```

Output:

Following are the arguments entered in the command line

C:\testproject.exe

apple

mango

Number of arguments entered are

3.

3. What are header files? Are functions declared or defined in header files ?

Functions and macros are declared in header files. Header files would be included in source files by the compiler at the time of compilation.

Header files are included in source code using #include directive.#include<some.h> includes all the declarations present in the header file 'some.h'.

A header file may contain declarations of sub-routines, functions, macros and also variables which we may want to use in our program. Header files help in reduction of repetitive code.

Syntax of include directive:

```
#include<stdio.h> //includes the header file stdio.h, standard input output header into the source code
```

Functions can be declared as well as defined in header files. But it is recommended only to declare functions and not to define in the header files. When we include a header file in our program we actually are including all the functions, macros and variables declared in it.

In case of pre-defined C standard library header files ex(stdio.h), the functions calls are replaced by equivalent binary code present in the pre-compiled libraries. Code for C

standard functions are linked and then the program is executed. Header files with custom names can also be created.

Program: Custom header files example

```
/******
```

```
Index: restaurant.h
```

```
*****/
```

```
int billAll(int food_cost, int tax, int tip);
```

```
/******
```

```
Index: restaurant.c
```

```
*****/
```

```
#include<stdio.h>
```

```
int billAll(int food_cost, int tax, int tip) {
```

```
int result;
```

```
result = food_cost + tax + tip;
```

```
printf("Total bill is %d\n",result);
```

```
return result;
```

```
}
```

Material from Interview Mantra. Subscribe to free updates via email.

```
/******
```

```
Index: main.c
```

```
*****/
```

```
#include<stdio.h>
```

```
#include"restaurant.h"
```

```
int main() {
```

```
int food_cost, tax, tip;
```

```
food_cost = 50;
```

```
tax = 10;
```

```
tip = 5;
```

```
billAll(food_cost,tax,tip);
```

```
return 0;
```

```
}
```

4. What are the differences between formal arguments and actual arguments of a function?

Argument: An argument is an expression which is passed to a function by its caller (or macro by its invoker) in order for the function(or macro) to perform its task. It is an expression in the comma-separated list bound by the parentheses in a function call expression.

Actual arguments:

The arguments that are passed in a function call are called actual arguments. These arguments are defined in the calling function.

Formal arguments:

The formal arguments are the parameters/arguments in a function declaration. The scope of formal arguments is local to the function definition in which they are used. Formal arguments belong to the called function. Formal arguments are a copy of the actual arguments. A change in formal arguments would not be reflected in the actual arguments.

Example:

```
#include <stdio.h>
void sum(int i, int j, int k);
/* calling function */
int main() {
    int a = 5;
    // actual arguments
    sum(3, 2 * a, a);
    return 0;
}
/* called function */
/* formal arguments*/
void sum(int i, int j, int k) {
    int s;
    s = i + j + k;
    printf("sum is %d", s);
}
```

Here 3,2*a,a are actual arguments and i,j,k are formal arguments.

5. What is pass by value in functions?

Pass by Value: In this method, the value of each of the actual arguments in the calling function is copied into corresponding formal arguments of the called function. In pass by value, the changes made to formal arguments in the called function have no effect on the values of actual arguments in the calling function.

Example:

```
#include <stdio.h>
void swap(int x, int y) {
    int t;
    t = x;
    x = y;
    y = t;
} int main() {
    int m = 10, n = 20;
    printf("Before executing swap m=%d n=%d\n", m, n);
    swap(m, n);
    printf("After executing swap m=%d n=%d\n", m, n);
    return 0;
}
```

Output:

Before executing swap m=10 n=20

After executing swap m=10 n=20

Explanation:

In the main function, value of variables m, n are not changed though they are passed to function 'swap'. Swap function has a copy of m, n and hence it can not manipulate the actual value of arguments passed to it.

6. What is pass by reference in functions?

Pass by Reference: In this method, the addresses of actual arguments in the calling function are copied into formal arguments of the called function. This means that using these addresses, we would have an access to the actual arguments and hence we would be able to manipulate them. C does not support Call by reference.

But it can be simulated using pointers.

Example:

```
#include <stdio.h>
/* function definition */
void swap(int *x, int *y) {
    int t;
    t = *x; /* assign the value at address x to t */
    *x = *y; /* put the value at y into x */
    *y = t; /* put the value at to y */
}
int main() {
    int m = 10, n = 20;
    printf("Before executing swap m=%d n=%d\n", m, n);
    swap(&m, &n);
    printf("After executing swap m=%d n=%d\n", m, n);
    return 0;
}
```

Output:

Before executing swap m=10 n=20

After executing swap m=20 n=10

Explanation:

In the main function, address of variables m, n are sent as arguments to the function 'swap'. As swap function has the access to address of the arguments, manipulation of passed arguments inside swap function would be directly reflected in the values of m, n.

7. What are the differences between getchar() and scanf() functions for reading strings?

Differences between getchar and scanf functions for reading strings:

scanf getchar

1. Entering of each character should be followed by return key. 1. Need not type return key.
2. Continuous stream of characters cannot be directly supplied using scanf function.
2. Continuous stream of characters can be directly supplied using getchar function
3. Scanf function can be used to provide data at execution time irrespective of its type(int, char, float).

Example:

```
#include<stdio.h>
int main() {
    char a[10];
    printf("Enter a: \n");
    scanf("%s",a);
}
```



```
return 0;
}
```

3. getchar() function is used only with character type.

Example:

```
#include<stdio.h>
int main() {
char a;
printf("Enter any character: \n");
a = getchar();
printf("Character entered:%c \n",a);
return 0;
}
```

4. scanf() returns the number of items read Successfully. A return value 0 indicates that no fields were read. EOF(end of file) is returned in case of an error or if end-of-file/end-of-string character is encountered.

4. getchar() returns the character entered as the value of the function. It returns EOF in case of an error. It is recommended to use getchar instead of scanf.

8. Out of the functions fgets() and gets(), which one is safer to use and why?

Out of functions fgets() and gets(), fgets() is safer to use. gets() receives a string from the keyboard and it is terminated only when the enter key is hit. There is no limit for the input string. The string can be too long and may lead to buffer overflow.

Example:

```
gets(s) /* s is the input string */
```

Whereas fgets() reads string with a specified limit, from a file and displays it on screen. The function fgets() takes three arguments.

First argument : address where the string is stored.

Second argument : maximum length of the string.

Third argument : pointer to a FILE.

Example:

```
fgets(s,20,fp); /* s: address of the string, 20: maximum length of string, fp: pointer to a file */
```

The second argument limits the length of string to be read. Thereby it avoids overflow of input buffer. Thus fgets() is preferable to gets().

9. What is the difference between the functions strdup() and strcpy()?

strcpy function: copies a source string to a destination defined by user. In strcpy function both source and destination strings are passed as arguments. User should make sure that destination has enough space to accommodate the string to be copied.

'strcpy' sounds like short form of "string copy".

Syntax:

```
strcpy(char *destination, const char *source);
```

Source string is the string to be copied and destination string is string into which source string is copied. If successful, strcpy subroutine returns the address of the copied string.

Otherwise, a null pointer is returned.

Example Program:

```
#include<stdio.h>
#include<string.h>
int main() {
char myname[10];
//copy contents to myname
strcpy(myname, "interviewmantra.net");
//print the string
puts(myname);
return 0;
}
```

Output:

interviewmantra.net

Explanation:

If the string to be copied has more than 10 letters, strcpy cannot copy this string into the string 'myname'. This is because string 'myname' is declared to be of size 10 characters only.

In the above program, string "nodalo" is copied in myname and is printed on output screen.

strdup function: duplicates a string to a location that will be decided by the function itself. Function will copy the contents of string to certain memory location and returns the address to that location. 'strdup' sounds like short form of "string duplicate"

Syntax:

strdup (const char *s);

strdup returns a pointer to a character or base address of an array. Function returns address of the memory location where the string has been copied. In case free space could not be created then it returns a null pointer.

Both strcpy and strdup functions are present in header file <string.h>

Program: Program to illustrate strdup().

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main() {
char myname[] = "interviewmantra.net";
//name is pointer variable which can store the address of memory location of string
char* name;
//contents of myname are copied in a memory address and are assigned to name
name = strdup(myname);
//prints the contents of 'name'
puts(name);
//prints the contents of 'myname'
puts(myname);
//memory allocated to 'name' is now freed
free(name);
}
```

```
return 0;
}
```

Output:

```
interviewmantra.net
interviewmantra.net
```

Explanation:

String myname consists of "interviewmantra.net" stored in it. Contents of myname are copied in a memory address and memory is assigned to name. At the end of the program, memory can be freed using free(name);

CHAPTER 4: Pointers**1. What is a pointer in C?**

A pointer is a special variable in C language meant just to store address of any other variable or function. Pointer variables unlike ordinary variables cannot be operated with all the arithmetic operations such as '*', '%' operators.

It follows a special arithmetic called as pointer arithmetic.

A pointer is declared as:

```
int *ap;
int a = 5;
```

In the above two statements an integer was declared and initialized to 5. A pointer to an integer with name ap was declared.

Next before ap is used

```
ap=&a;
```

This operation would initialize the declared pointer to int. The pointer ap is now said to point to a.

Operations on a pointer:

- **Dereferencing operator ' * ':** This operator gives the value at the address pointed by the pointer. For example after the above C statements if we give

```
printf("%d",*ap);
```

Actual value of a that is 5 would be printed. That is because ap points to a.

- **Addition operator ' + ':** Pointer arithmetic is different from ordinary arithmetic.

Material from Interview Mantra. Subscribe to free updates via email.

```
ap=ap+1;
```

Above expression would not increment the value of ap by one, but would increment it by the number of bytes of the data type it is pointing to. Here ap is pointing to an integer variable hence ap is incremented by 2 or 4 bytes depending upon the compiler.

2. What are the advantages of using pointers?

Pointers are special variables which store address of some other variables.

Syntax: datatype *ptr;

Here * indicates that ptr is a pointer variable which represents value stored at a particular address.

Example: int *p;

'p' is a pointer variable pointing to address location where an integer type is stored.

Advantages:

1. Pointers allow us to pass values to functions using *call by reference*. This is useful when large sized arrays are passed as arguments to functions. A function can return more than one

value by using *call by reference*.

2. Dynamic allocation of memory is possible with the help of pointers.

3. We can resize data structures. For instance, if an array's memory is fixed, it cannot be resized. But in case of an array whose memory is created out of malloc can be resized.

4. Pointers point to physical memory and allow quicker access to data.

3. What are the differences between malloc() and calloc()?

Allocation of memory at the time of execution is called dynamic memory allocation. It is done using the standard library functions malloc() and calloc(). It is defined in "stdlib.h".

malloc(): used to allocate required number of bytes in memory at runtime. It takes one argument, viz. size in bytes to be allocated.

Syntax:

```
void * malloc(size_t size);
```

Example:

```
a = (int*) malloc(4);
```

4 is the size (in bytes) of memory to be allocated.

calloc(): used to allocate required number of bytes in memory at runtime. It needs *two arguments* viz.,

1. total number of data and

2. size of each data.

Syntax:

```
void * calloc(size_t nmemb, size_t size);
```

Example:

```
a = (int*) calloc(8, sizeof(int));
```

Here sizeof indicates the size of the data type and 8 indicates that we want to reserve space for storing 8 integers.

Differences between malloc() and calloc() are:

1. Number of arguments differ.

2. By default, memory allocated by malloc() contains garbage values. Whereas memory allocated by calloc() contains all zeros.

4. How to use realloc() to dynamically increase size of an already allocated array?

realloc(): This function is used to increase or decrease the size of any dynamic memory which is allocated using malloc() or calloc() functions.

Syntax: void *realloc(void *ptr, size_t newsize);

The first argument 'ptr' is a pointer to the memory previously allocated by the malloc or calloc functions. The second argument 'newsize' is the size in bytes, of a new memory region to be allocated by realloc. This value can be larger or smaller than the previously allocated memory. The realloc function adjusts the old memory region if newsize is smaller than the size of old memory.

If the newsize is larger than the existing memory size, it increases the size by copying the contents of old memory region to new memory region. The function then deallocates the old memory region. realloc function is helpful in managing a dynamic array whose size may change during execution.