

Programs on 1D Arrays

- 1) WAP in C to create two arrays viz. integer array and float array. Your program must initialize the integer array to the values 1, 2, 3, 5, 6, 9, and 10 respectively. Similarly, it must initialize float array to the values 1.90, 2.98, 5.57, and 6.67 respectively. Finally display these initialized values.

Answer:

```
#include<stdio.h>

int main()
{
    int i;
    int a[] = {1,2,3,5,6,9,10};
    float b[] = {1.90, 2.98,5.57, 6.67};
    printf("Printing integer array:\n");
    for(i=0;i<7;i++)
    {
        printf("%d ",a[i]);
    }
    printf("\nPrinting floating array:\n");
    for(i=0;i<4;i++)
    {
        printf("%f ",b[i]);
    }
    return 0;
}
```

Sample Input/Output:

```
Printing integer array:
1 2 3 5 6 9 10
Printing floating array:
1.900000 2.980000 5.570000 6.670000
```

- 2) WAP in C to input n numbers from the user and display the entered numbers using the concept of 1D array.

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("Entered numbers are:\n");
    for(i=0;i<n;i++)
```

```

{
printf("%d ",a[i]);
}
return 0;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Entered numbers are:
24 63 -5 8 1

```

3) WAP in C to calculate the sum and average of n numbers entered by the user using array.

Answer:

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

```

int main()
{
float a[100],sum=0.00,avg;
int n,i;
printf("Enter n:");
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%f",&a[i]);
sum=sum+a[i];
}
avg=sum/n;
printf("Sum = %f and average =%f",sum,avg);
return 0;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
1 2 3 4 5
Sum = 15.000000 and average =3.000000

```

4) WAP in C to find greatest number among 10 numbers entered by the user

Answer:

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

```

int main()
{
int a[10],i,great;
printf("Enter ten numbers:\n");
for(i=0;i<10;i++)

```

```

{
scanf("%d",&a[i]);
}
great=a[0];
for(i=1;i<10;i++)
{
if(a[i]>great)
    great=a[i];
}
printf("The greatest number is %d",great);
return 0;
}

```

Sample Input/Output:

```

Enter ten numbers:
24 63 -5 8 1 33 56 29 17 -10
The greatest number is 63

```

5) WAP in C to find a greatest number among n numbers entered by the user

Answer:

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

```

int main()
{
int a[100],n,i,great;
printf("Enter n:");
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
great=a[0];
for(i=1;i<n;i++)
{
if(a[i]>great)
    great=a[i];
}
printf("The greatest number is %d",great);
return 0;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
The greatest number is 63

```

6) WAP in C to find a least number among 10 numbers entered by the user

Answer:

```
#include<stdio.h>

int main()
{
    int a[10],i,least;
    printf("Enter ten numbers:\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&a[i]);
    }
    least=a[0];
    for(i=1;i<10;i++)
    {
        if(a[i]< least)
            least=a[i];
    }
    printf("The smallest number is %d",least);
    return 0;
}
```

Sample Input/Output:

```
Enter ten numbers:
24 63 -5 8 1 33 56 29 17 -10
The smallest number is -10
```

7) WAP in C to find a least number among n numbers entered by the user

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i,least;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    least=a[0];
    for(i=1;i<n;i++)
    {
        if(a[i]< least)
            least=a[i];
    }
    printf("The smallest number is %d",least);
    return 0;
}
```

```
}
```

Sample Input/Output:

```
Enter n:5
Start entering values...
24 63 -5 8 1
The smallest number is -5
```

8) WAP in C to sort n numbers entered by the user in ascending order.

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i,j,temp;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("In ascending order:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    return 0;
}
```

Sample Input/Output:

```
Enter n:5
Start entering values...
24 63 -5 8 1
In ascending order:
-5 1 8 24 63
```

9) WAP in C to sort n numbers entered by the user in descending order.

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i,j,temp;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]<a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("In descending order:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    return 0;
}
```

Sample Input/Output:

```
Enter n:5
Start entering values...
24 63 -5 8 1
In descending order:
63 24 8 1 -5
```

10) WAP in C to find the smallest, largest and 3rd largest number from a list of n numbers entered by the user. Use ascending sort.

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i,j,temp;
```

```

printf("Enter n:");
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(a[i]>a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
}
printf("Smallest=%d\n",a[0]);
printf("Largest=%d\n",a[n-1]);
printf("Third Largest=%d",a[n-3]);
return 0;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Smallest=-5
Largest=63
Third Largest=8

```

11) WAP in C to find the smallest, largest and 3rd largest number from a list of n numbers entered by the user. Use descending sort.

Answer:

```

#include<stdio.h>

int main()
{
int a[100],n,i,j,temp;
printf("Enter n:");
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)

```

```

{
    for(j=i+1;j<n;j++)
    {
        if(a[i]<a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
}
printf("Smallest=%d\n",a[n-1]);
printf("Largest=%d\n",a[0]);
printf("Third Largest=%d",a[2]);
return 0;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Smallest=-5
Largest=63
Third Largest=8

```


Programs on 2D Arrays

1. WAP in C to initialize and display a 3×4 matrix.

Answer:

```
#include<stdio.h>

int main()
{
    int a[3][4]={1,2,3,4,5,6,7,8,9,1,2,4};
    int i,j;
    printf("Initialized matrix is:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
        {
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Sample Input/Output:

```
Initialized matrix is:
1 2 3 4
5 6 7 8
9 1 2 4
```

2. WAP in C to input and display 3×3 matrix entered by the user.

Answer:

```
#include<stdio.h>

int main()
{
    int a[3][3],i,j;
    printf("Enter a matrix:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Entered matrix is:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d ",a[i][j]);
        }
    }
}
```

```
printf("\n");
}
return 0;
}
```

Sample Input/Output:

```
Enter a matrix:
1 2 3
4 5 6
7 8 9
Entered matrix is:
1 2 3
4 5 6
7 8 9
```

3. WAP in C to input and display $m \times n$ matrix entered by the user

Answer:

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

```
int main()
{
    int a[10][10],i,j,m,n;
    printf("Enter the order:");
    scanf("%d%d",&m,&n);
    printf("Enter a matrix:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Entered matrix is:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("%d ",a[i][j]);
        }
    }
    printf("\n");
}
return 0;
}
```

Sample Input/Output:

```
Enter the order:3 2
Enter a matrix:
1 2
3 4
5 6
Entered matrix is:
1 2
3 4
5 6
```

4. WAP in C to display the transpose of a $m \times n$ matrix entered by the user

Answer:

```
#include<stdio.h>

int main()
{
    int a[10][10],i,j,m,n;
    printf("Enter the order:");
    scanf("%d%d",&m,&n);
    printf("Enter a matrix:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Transposed matrix is:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            printf("%d ",a[j][i]);
        }
    }
    printf("\n");
}
return 0;
}
```

Sample Input/Output:

```
Enter the order:3 3
Enter a matrix:
1 2 3
4 5 6
7 8 9
Transposed matrix is:
1 4 7
2 5 8
3 6 9
```

```
Enter the order:3 2
Enter a matrix:
1 2
3 4
5 6
Transposed matrix is:
1 3 5
2 4 6
```

```
Enter the order:1 3
Enter a matrix:
1 2 3
Transposed matrix is:
1
2
3
```

5. WAP in C to display the sum and difference matrices of two 3×3 matrices entered by the user

Answer:

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

```
int main()
{
```

```

int a[3][3],b[3][3],c[3][3],d[3][3],i,j;
printf("Enter first matrix:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        scanf("%d",&a[i][j]);
    }
}
printf("Enter second matrix:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        scanf("%d",&b[i][j]);
    }
}
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        c[i][j]=a[i][j]+b[i][j];
        d[i][j]=a[i][j]-b[i][j];
    }
}
printf("Sum matrix is:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
printf("Difference matrix is:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d ",d[i][j]);
    }
    printf("\n");
}
return 0;
}

```

Sample Input/Output:

```
Enter first matrix:
1 1 1
1 1 1
1 1 1
Enter second matrix:
1 1 1
1 1 1
1 1 1
Sum matrix is:
2 2 2
2 2 2
2 2 2
Difference matrix is:
0 0 0
0 0 0
0 0 0
```

6. WAP in C to display the product of two 3×3 matrices entered by the user

Answer:

```
#include<stdio.h>

int main()
{
    int a[3][3],b[3][3],c[3][3],i,j,k,sum;
    printf("Enter first matrix:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter second matrix:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            sum=0;
            for(k=0;k<3;k++)
            {
                sum=sum+a[i][k]*b[k][j];
            }
            c[i][j]=sum;
        }
    }
}
```

```

}
printf("Product matrix is:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
return 0;
}

```

Sample Input/Output:

```

Enter first matrix:
1 2 3
4 5 6
7 8 9
Enter second matrix:
1 1 1
2 2 2
3 3 3
Product matrix is:
14 14 14
32 32 32
50 50 50

```

7. WAP in C to find the product of two matrices of different orders entered by the user.
Display suitable message if any incompatibility is detected.

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

```

int main()
{
    int a[10][10],b[10][10],c[10][10],i,j,k,sum;
    int m1,n1,m2,n2;
    printf("Enter order of first matrix:");
    scanf("%d%d",&m1,&n1);
    printf("Enter order of second matrix:");
    scanf("%d%d",&m2,&n2);
    if(n1!=m2)
    {
        printf("Product not compatible!");
        goto skip;
    }
    printf("Enter first matrix:\n");
    for(i=0;i<m1;i++)
    {
        for(j=0;j<n1;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter second matrix:\n");

```

```

for(i=0;i<m2;i++)
{
    for(j=0;j<n2;j++)
    {
        scanf("%d",&b[i][j]);
    }
}
for(i=0;i<m1;i++)
{
    for(j=0;j<n2;j++)
    {
        sum=0;
        for(k=0;k<n1;k++)
        {
            sum=sum+a[i][k]*b[k][j];
        }
        c[i][j]=sum;
    }
}
printf("Product matrix is:\n");
for(i=0;i<m1;i++)
{
    for(j=0;j<n2;j++)
    {
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
skip:
return 0;
}

```

Sample Input/Output:

<pre> Enter order of first matrix:3 2 Enter order of second matrix:2 3 Enter first matrix: 1 2 3 4 5 6 Enter second matrix: 1 2 3 4 5 6 Product matrix is: 9 12 15 19 26 33 29 40 51 </pre>	<pre> Enter order of first matrix:3 3 Enter order of second matrix:3 3 Enter first matrix: 1 2 3 4 5 6 7 8 9 Enter second matrix: 1 1 1 2 2 2 3 3 3 Product matrix is: 14 14 14 32 32 32 50 50 50 </pre>
---	--

8. WAP in C to find largest number in a $m \times n$ matrix entered by the user. Also replace all the principle diagonal elements by it.

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

```

int main()
{
    int a[10][10],i,j,great,m,n;
    printf("Enter the order:");
    scanf("%d%d",&m,&n);
    printf("Enter a matrix:\n");
}

```

```

for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        scanf("%d",&a[i][j]);
    }
}
great=a[0][0];
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        if(a[i][j]>great)
            great=a[i][j];
    }
}
printf("Greatest number=%d\n",great);
printf("Required matrix is:\n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        if(i==j)
            a[i][j]=great;
        printf("%d ",a[i][j]);
    }
}
printf("\n");
}
return 0;
}

```

Sample Input/Output:

```

Enter the order:3 3
Enter a matrix:
1 2 3
4 5 6
7 8 3
Greatest number=8
Required matrix is:
8 2 3
4 8 6
7 8 8

```

9. WAP in C to find the sum of diagonal elements in a $m \times n$ matrix entered by the user.

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

```

int main()
{
    int a[10][10],i,j,m,n,sum=0;
    printf("Enter the order:");
}

```



```

scanf("%d%d",&m,&n);
printf("Enter a matrix:\n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        scanf("%d",&a[i][j]);
        if(i==j || (i+j)==(n-1))
            sum=sum+a[i][j];
    }
}
printf("Sum of diagonal elements=%d",sum);
return 0;
}

```

Sample Input/Output:

```

Enter the order:3 3
Enter a matrix:
1 2 3
4 5 6
7 8 9
Sum of diagonal elements=25

```

Programs on Strings

1) WAP in C to input a string from the user and display the entered string

Answer:

```
#include<stdio.h>

int main()
{
char str[30];
printf("Enter a string:");
scanf("%[^\n]s",str);
printf("Entered string is %s",str);
return 0;
}
```

Sample Input/Output:

```
Enter a string:Tribhuvan University
Entered string is Tribhuvan University
```

OR

```
#include<stdio.h>

int main()
{
char str[30];
printf("Enter a string:");
gets(str);
printf("Entered string is %s",str);
return 0;
}
```

Sample Input/Output:

```
Enter a string:Tribhuvan University
Entered string is Tribhuvan University
```

2) WAP in C to display the ASCII value of a character entered by the user

Answer:

```
#include<stdio.h>

int main()
{
char ch;
printf("Enter a character:");
ch=getchar();
printf("ASCII value of %c = %d",ch,ch);
return 0;
}
```

Sample Input/Output:

```
Enter a character:A
ASCII value of A = 65
```

3) WAP in C to initialize a string and display the initialized string

Answer:

```
#include<stdio.h>

int main()
{
    char str1[]="Pokhara";
    char str2[]={ 'P','o','k','h','a','r','a','\0' };
    printf("Initialized strings are:\n");
    printf("%s and\n",str1);
    printf("%s",str2);
    return 0;
}
```

Sample Input/Output:

```
Initialized strings are:
Pokhara and
Pokhara
```

4) WAP in C to input and display a string by using character array method. i.e. by accessing individual character element of a string variable.

Answer:

```
#include<stdio.h>

int main()
{
    char str[30];
    int i=0;
    printf("Enter a string:");
    while((str[i]=getchar())!='\n')
    {
        i++;
    }
    str[i]='\0';
    i=0;
    printf("Entered string is:");
    while(str[i]!='\0')
    {
        putchar(str[i]);
        i++;
    }
    return 0;
}
```

Sample Input/Output:

```
Enter a string:Tribhuvan University
Entered string is:Tribhuvan University
```

5) WAP in C to count the no. of characters and words in a line of string entered by the user.

Answer:

```
#include<stdio.h>

int main()
{
    char str[100];
    int i=0,nchars,nwords=1;
    printf("Enter a string:");
    gets(str);
    while(str[i]!='\0')
    {
        if(str[i]==' ')
            nwords++;
        i++;
    }
    nchars=i;
    printf("No. of chars=%d\n",nchars);
    printf("No. of words=%d",nwords);
    return 0;
}
```

Sample Input/Output:

```
Enter a string:The quick brown fox jumps over the lazy dog
No. of chars=43
No. of words=9
```

Programs on Strings using library functions

1) WAP in C to find the length of a string entered by the user using library function

Answer:

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

int main()
{
    int length;
    char str[20];
    printf("Enter a string:");
    scanf("%s",str);
    length=strlen(str);
    printf("The length of entered string is %d",length);
    return 0;
}
```

Sample Input/Output:

```
Enter a string:Nepal
The length of entered string is 5
```

2) WAP in C to copy a string entered by the user.

Answer:

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

int main()
{
    char str1[30],str2[30];
    printf("Enter a string:");
    scanf("%s",&str1);
    strcpy(str2,str1);
    printf("The copied string is:%s",str2);
    return 0;
}
```

Sample Input/Output:

```
Enter a string:Nepal
The copied string is:Nepal
```

3) WAP in C to concatenate two strings entered by the user using the library function.

Answer:

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

int main()
{
    char str1[20];
    char str2[20];
    printf("Enter first string:");
    scanf("%s",str1);
    printf("Enter second string:");
    scanf("%s",str2);
    strcat(str1,str2);
    printf("The concatenated string is: %s",str1);
    return 0;
}
```

Sample Input/Output:

```
Enter first string:Tribhuvan
Enter second string:University
The concatenated string is: TribhuvanUniversity
```

4) WAP in C to reverse a string entered by the user using library function

Answer:

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

int main()
{
    char str[30],rstr[30];
    printf("Enter a string:");
    scanf("%s",&str);
```

```

strcpy(rstr, str);
strrev(rstr);
printf("The reversed string is:%s", rstr);
return 0;
}

```

Sample Input/Output:

```

Enter a string:Pokhara
The reversed string is:arahkoP

```

5) WAP in C to check a string entered by the user is Palindrome or not by using the library function

Answer:

```

#include<stdio.h>
#include<string.h>

int main()
{
char str[30], rstr[30];
printf("Enter a string to test: ");
scanf("%s", str);
strcpy(rstr, str);
strrev(rstr);
if(strcmp(str, rstr)==0)
printf("Palindrome");
else
printf("Not palindrome");
return 0;
}

```

Sample Input/Output:

Sample Run1:

```

Enter a string to test: madam
Palindrome

```

Sample Run2:

```

Enter a string to test: Nepal
Not palindrome

```

6) WAP in C to compare two string entered by the user.

```

#include<stdio.h>

```

```

int main()
{
char str1[30], str2[30];
int value;
printf("Enter first string:");
gets(str1);
printf("Enter second string:");
gets(str2);

```

```

value=strcmp(str1,str2);
if(value==0)
    printf("str1=str2");
else if(value>0)
    printf("str1>str2");
else
    printf("str1<str2");
return 0;
}

```

Sample Input/Output:

Sample Run 1:

```

Enter first string:Ram
Enter second string:Anish
str1>str2

```

Sample Run 2:

```

Enter first string:Anish
Enter second string:Ram
str1<str2

```

Sample Run 3:

```

Enter first string:Nepal
Enter second string:Nepal
str1=str2

```

7) WAP in C to sort names of n employees in ascending order (Alphabetical order)

Answer:

```

#include<stdio.h>
#include<string.h>

int main()
{
    char empname[20][20],temp[20];
    int n,i,j;
    printf("Enter number of employees:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        gets(empname[i]);
    }
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(strcmp(empname[i],empname[j])>0)
            {
                strcpy(temp,empname[j]);
                strcpy(empname[j],empname[i]);
                strcpy(empname[i],temp);
            }
        }
    }
}

```

```

}
}
printf("The entered names in ascending order is:\n");
for(i=0;i<n;i++)
{
puts(empname[i]);
}
return 0;
}

```

Sample Input/Output:

```

Enter number of employees:5
Start entering names...
Ramesh
Dinesh
Anish
Hari
Bikash
In Alphabetical Order:
Anish
Bikash
Dinesh
Hari
Ramesh

```

Additional Programs on Strings

1) WAP in C to convert a string entered by the user from lower case to upper case.

Answer:

```

#include<stdio.h>
#include<string.h>

int main()
{
int i;
char str[30];
printf("Enter a string:");
gets(str);
for(i=0;i<strlen(str);i++)
{
str[i]=str[i]-32;
}
printf("In UPPER CASE=%s",str);
return 0;
}

```

Sample Input/Output:

```

Enter a string:nepal
In UPPER CASE=NEPAL

```


2) WAP in C to convert a string entered by the user from upper case to lower case

Answer:

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

int main()
{
    int i;
    char str[30];
    printf("Enter a string:");
    gets(str);
    for(i=0;i<strlen(str);i++)
    {
        str[i]=str[i]+32;
    }
    printf("In lower case=%s",str);
    return 0;
}
```

Sample Input/Output:

```
Enter a string:NEPAL
In lower case=nepal
```

3) WAP in C to convert a string entered by the user. While converting each character of the entered string, it must be converted to UPPER CASE if it is in lower case and vice versa.

Answer:

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

int main()
{
    int i;
    char str[30];
    printf("Enter a string:");
    gets(str);
    for(i=0;i<strlen(str);i++)
    {
        if(str[i]>=65 && str[i]<=90)
            str[i]=str[i]+32;
        else
            str[i]=str[i]-32;
    }
    printf("Required String=%s",str);
    return 0;
}
```

Sample Input/Output:

```
Enter a string:NePaL
Required String=nEpAl
```

Pattern Questions on Strings

- 1) Write a C program to display following pattern without formatted input/output statements.

Programming
rogrammin
ogrammi
gramm
ram
a

Answer:

```
#include<stdio.h>

int main()
{
    int i,j,k=0;
    char str[]="programming";
    for(i=10;i>=1;i--)
    {
        for(j=k;j<=i;j++)
            putchar(str[j]);
        putchar('\n');
        k++;
    }
    return 0;
}
```

- 2) Write a C program to display following pattern using unformatted input/output statements.

```
      *
    * * *
  * * * * *
* * * * * * *
* * * * * * *
  * * * * *
    * * *
      *
```

Answer:

```
#include<stdio.h>

int main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=5;j>i;j--)
            putchar(' ');
        for(j=1;j<=2*i-1;j++)
            putchar('*');
    }
}
```

```
putchar('\n');
}
for(i=4;i>=1;i--)
{
    for(j=5;j>i;j--)
        putchar(' ');
    for(j=1;j<=2*i-1;j++)
        putchar('*');
    putchar('\n');
}
return 0;
}
```

Basic Programs using Functions

- 1) WAP in C to find the square of a number entered by the user using the concept of a user defined function.

Answer:

```
#include<stdio.h>

float square(float); /*Function Prototype*/

int main()
{
float x, sq;
printf("Enter a number:");
scanf("%f",&x);
sq = square(x); /*Function Call*/
printf("The square of %f is %f",x,sq);
return 0;
}

float square(float p)
{
return p*p;
}
```

Sample Input/Output:-

```
Enter a number:1.5
The square of 1.500000 is 2.250000
```

- 2) Write a program in C to display the sum of two numbers entered by the user using the concept of function with return type and argument.

Answer:

```
#include<stdio.h>

int addition(int,int);

int main()
{
int a,b,sum;
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
sum=addition(a,b);
printf("The sum is %d",sum);
return 0;
}
```

```
int addition(int p,int q)
{
    return p+q;
}
```

Sample Input/Output:

```
Enter two numbers:2 3
The sum is 5
```

3) *Write a program in C to display the sum of two numbers entered by the user using the concept of function with return type and no argument.*

Answer:

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

```
int addition();
```

```
int main()
{
    int sum;
    sum=addition();
    printf("The sum is %d",sum);
    return 0;
}
```

```
int addition()
{
    int a,b;
    printf("Enter two numbers:");
    scanf("%d%d",&a,&b);
    return a+b;
}
```

Sample Input/Output:

```
Enter two numbers:2 3
The sum is 5
```

4) *Write a program in C to display the sum of two numbers entered by the user using the concept of function with no return type and argument.*

Answer:

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

```
void addition(int,int);
```

```
int main()
{
```

```

int a,b;
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
addition(a,b);
return 0;
}

```

```

void addition(int p,int q)
{
int sum;
sum=p+q;
printf("The sum is %d",sum);
}

```

Sample Input/Output:

```

Enter two numbers:2 3
The sum is 5

```

5) Write a program in C to display the sum of two numbers entered by the user using the concept of function with no return type and no argument.

Answer:

```

#include<stdio.h>

```

```

void addition();

```

```

int main()
{
addition();
return 0;
}

```

```

void addition(int p,int q)
{
int a,b,sum;
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
sum=a+b;
printf("The sum is %d",sum);
}

```

Sample Input/Output:

```

Enter two numbers:2 3
The sum is 5

```

6) Write a program in C to check whether a number entered by the user is prime or composite using the concept of a user defined function.

Answer:

```
#include<stdio.h>

int isPrime(int);

int main()
{
    int n;
    printf("Enter a no.:");
    scanf("%d",&n);
    if(isPrime(n))
        printf("Prime!");
    else
        printf("Composite!");
    return 0;
}

int isPrime(int n)
{
    int i;
    for(i=2;i<n;i++)
    {
        if(n%i==0)
            return 0;
    }
    return 1;
}
```

Sample Input/Output:

Sample Run1:

```
Enter a no.:7
Prime!
```

Sample Run2:

```
Enter a no.:6
Composite!
```

7) Write a program in C to display the prime numbers in the range entered by the user.

Answer:

```
#include<stdio.h>

int isPrime(int);
```

```

int main()
{
    int i,l,u;
    printf("Enter a range:");
    scanf("%d%d",&l,&u);
    for(i=l;i<=u;i++)
    {
        if(isPrime(i))
            printf("%d ",i);
    }
    return 0;
}

```

```

int isPrime(int n)
{
    int i;
    for(i=2;i<n;i++)
    {
        if(n%i==0)
            return 0;
    }
    return 1;
}

```

Sample Input/Output:

```

Enter a range:2 50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

```

8) Write a program in C to check whether a number entered by the user is Armstrong or not. Assume the case of three digit Armstrong no.

Answer:

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

```
int isArmstrong(int);
```

```

int main()
{
    int n;
    printf("Enter a three digit no.:");
    scanf("%d",&n);
    if(isArmstrong(n))
        printf("Armstrong!");
    else
        printf("Not Armstrong!");
}

```



```

return 0;
}

int isArmstrong(int n)
{
    int a,r,s=0;
    a=n;
    while(n>0)
    {
        r=n%10;
        s=s+r*r*r;
        n=n/10;
    }
    if(s==a)
        return 1;
    else
        return 0;
}

```

Sample Input/Output:

Sample Run1:

```

Enter a three digit no.:153
Armstrong!

```

Sample Run2:

```

Enter a three digit no.:154
Not Armstrong!

```

9) Write a program in C to display the Armstrong numbers in a range entered by the user.
Assume the case of three digit Armstrong no.

Answer:

```

#include<stdio.h>

int isArmstrong(int);

int main()
{
    int i,l,u;
    printf("Enter a range:");
    scanf("%d%d",&l,&u);
    for(i=l;i<=u;i++)
    {
        if(isArmstrong(i))
            printf("%d ",i);
    }
    return 0;
}

```

```

}

int isArmstrong(int n)
{
    int a,r,s=0;
    a=n;
    while(n>0)
    {
        r=n%10;
        s=s+r*r*r;
        n=n/10;
    }
    if(s==a)
        return 1;
    else
        return 0;
}

```

Sample Input/Output:

```

Enter a range:100 999
153 370 371 407

```

10) WAP in C to convert a decimal no. into its binary equivalent using the concept of a user defined function.

Answer:

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

```
int dectobin(int);
```

```

int main()
{
    int n,value;
    printf("Enter a decimal no.:");
    scanf("%d",&n);
    value=dectobin(n);
    printf("Equivalent Binary=%d",value);
    return 0;
}

```

```

int dectobin(int n)
{
    int r,i=1,s=0;
    while(n>0)
    {
        r=n%2;

```

```

s=s+r*i;
i=i*10;
n=n/2;
}
return s;
}

```

Sample Input/Output:

```

Enter a decimal no.:14
Equivalent Binary=1110

```

Programs on Pass by Value and Pass by Reference

11) Write a program in C to illustrate the concept of pass by value (or call by value)

Answer:

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

```
void swap(int,int);
```

```

int main()
{
int a,b;
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
printf("Before calling swap a=%d,b=%d\n",a,b);
swap(a,b);
printf("After calling swap a=%d,b=%d\n",a,b);
return 0;
}

```

```

void swap(int p,int q)
{
int temp;
temp=p;
p=q;
q=temp;
}

```

Sample Input/Output:

```

Enter two numbers:2 3
Before calling swap a=2,b=3
After calling swap a=2,b=3

```

12) Write a program in C to illustrate the concept of pass by reference (or call by reference)

Answer:

```

#include<stdio.h>

void swap(int *,int *);

int main()
{
    int a,b;
    printf("Enter two numbers:");
    scanf("%d%d",&a,&b);
    printf("Before calling swap a=%d,b=%d\n",a,b);
    swap(&a,&b);
    printf("After calling swap a=%d,b=%d\n",a,b);
    return 0;
}

void swap(int *p,int *q)
{
    int temp;
    temp=*p;
    *p=*q;
    *q=temp;
}

```

Sample Input/Output:

```

Enter two numbers:2 3
Before calling swap a=2,b=3
After calling swap a=3,b=2

```

Programs on passing 1D array to user defined function

13) Write a program in C to input ten numbers from the user and display the entered numbers using the concept of passing 1D array to a user defined function.

Answer:

```

#include<stdio.h>

void display(int []);

int main()
{
    int a[10],i;
    printf("Enter ten numbers...\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&a[i]);
    }
}

```

```

display(a);
return 0;
}

void display(int p[])
{
    int i;
    printf("Entered numbers are:\n");
    for(i=0;i<10;i++)
    {
        printf("%d ",p[i]);
    }
}

```

Sample Input/Output:

```

Enter ten numbers...
24 63 -5 8 1 54 56 87 98 12
Entered numbers are:
24 63 -5 8 1 54 56 87 98 12

```

14) Write a program in C to input n numbers from the user and display the entered numbers using the concept of passing 1D array to a user defined function.

Answer:

```

#include<stdio.h>

void display(int [],int);

int main()
{
    int a[100],n,i;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    display(a,n);
    return 0;
}

void display(int p[],int n)
{
    int i;
    printf("Entered numbers are:\n");
    for(i=0;i<n;i++)

```

```

{
printf("%d ",p[i]);
}
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Entered numbers are:
24 63 -5 8 1

```

15) Write a program in C to sort ten numbers entered by the user using the concept of passing 1D array to a user defined function.

Answer:

```

#include<stdio.h>

void sort(int []);

int main()
{
int a[10],i;
printf("Enter ten numbers\n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
sort(a);
printf("In Ascending order:\n");
for(i=0;i<10;i++)
{
printf("%d ",a[i]);
}
return 0;
}

void sort(int p[])
{
int i,j,temp;
for(i=0;i<9;i++)
{
for(j=i+1;j<10;j++)
{
if(p[i]>p[j])
{
temp=p[i];

```

```

        p[i]=p[j];
        p[j]=temp;
    }
}
}

```

Sample Input/Output:

```

Enter ten numbers
10 9 8 7 6 5 4 3 2 1
In Ascending order:
1 2 3 4 5 6 7 8 9 10

```

16) Write a program in C to sort n numbers entered by the user using the concept of passing 1D array to a user defined function.

Answer:

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

```
void sort(int [],int);
```

```

int main()
{
    int a[100],n,i;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    sort(a,n);
    printf("In Ascending order:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",a[i]);
    }
    return 0;
}

```

```

void sort(int p[],int n)
{
    int i,j,temp;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {

```

```

        if(p[i]>p[j])
        {
            temp=p[i];
            p[i]=p[j];
            p[j]=temp;
        }
    }
}
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
In Ascending order:
-5 1 8 24 63

```

17) Write a program in C to input ten numbers from the user and display the largest, third largest and smallest among those numbers using the concept of passing 1D array to a user defined function.

Answer:

```

#include<stdio.h>

void sort(int []);

int main()
{
    int a[10],i;
    printf("Enter ten numbers:\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&a[i]);
    }
    sort(a);
    printf("Largest=%d\n",a[9]);
    printf("Third Largest=%d\n",a[7]);
    printf("Smallest=%d",a[0]);
    return 0;
}

void sort(int p[])
{
    int i,j,temp;
    for(i=0;i<9;i++)
    {

```



```

        for(j=i+1;j<10;j++)
        {
            if(p[i]>p[j])
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
            }
        }
    }
}

```

Sample Input/Output:

```

Enter ten numbers:
24 63 -5 8 1 45 56 29 52 14
Largest=63
Third Largest=52
Smallest=-5

```

18) Write a program in C to input n numbers from the user and display the largest, third largest and smallest among those numbers using the concept of passing 1D array to a user defined function.

Answer:

```

#include<stdio.h>

void sort(int [],int);

int main()
{
    int a[100],n,i;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    sort(a,n);
    printf("Largest=%d\n",a[n-1]);
    printf("Third Largest=%d\n",a[n-3]);
    printf("Smallest=%d",a[0]);
    return 0;
}

void sort(int p[],int n)
{

```

```

int i,j,temp;
for(i=0;i<n-1;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(p[i]>p[j])
        {
            temp=p[i];
            p[i]=p[j];
            p[j]=temp;
        }
    }
}
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Largest=63
Third Largest=8
Smallest=-5

```

19) Write a program in C to input n numbers from the user inside the main() function. Pass these numbers to a user defined function. Find the largest value inside the user defined function. Return the largest value to the main() function and print it.

Answer:

```

#include<stdio.h>

int findLargest(int [],int);

int main()
{
    int a[100],n,i,great;
    printf("Enter n:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    great=findLargest(a,n);
    printf("Largest=%d",great);
    return 0;
}

```

```

int findLargest(int p[],int n)
{
    int i,great;
    great=p[0];
    for(i=1;i<n;i++)
    {
        if(p[i]>great)
            great=p[i];
    }
    return great;
}

```

Sample Input/Output:

```

Enter n:5
Start entering values...
24 63 -5 8 1
Largest=63

```

Programs on passing 2D arrays to user defined functions

20) Write a program in C to input a 3 x 3 matrix from the user and display the entered matrix using the concept of passing 2D array to a user defined function.

Answer:

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

```
void display(int [3][3]);
```

```

int main()
{
    int a[3][3],i,j;
    printf("Enter a 3 x 3 matrix:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    display(a);
    return 0;
}

```

```

void display(int p[3][3])
{
    int i,j;

```

```

printf("Entered matrix is:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d ",p[i][j]);
    }
    printf("\n");
}
}

```

Sample Input/Output:

```

Enter a 3 x 3 matrix:
1 2 3
4 5 6
7 8 9
Entered matrix is:
1 2 3
4 5 6
7 8 9

```

21) Write a program in C to input a $m \times n$ matrix from the user and display the entered matrix using the concept of passing 2D array to a user defined function.

Answer:

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

```
void display(int [10][10],int,int);
```

```

int main()
{
    int a[10][10],i,j,m,n;
    printf("Enter the order mxn:");
    scanf("%d%d",&m,&n);
    printf("Enter the matrix:\n");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    display(a,m,n);
    return 0;
}

```

```
void display(int p[10][10],int m,int n)
```

```

{
int i,j;
printf("Entered matrix is:\n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        printf("%d ",p[i][j]);
    }
printf("\n");
}
}

```

Sample Input/Output:

```

Enter the order mxn:3 3
Enter the matrix:
1 2 3
4 5 6
7 8 9
Entered matrix is:
1 2 3
4 5 6
7 8 9

```

22) Write a program in C to display the transpose of a $m \times n$ matrix entered by the user by using the concept of passing 2D array to a user defined function.

Answer:

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

```
void displayTranspose(int [10][10],int,int) ;
```

```

int main()
{
int a[10][10],i,j,m,n;
printf("Enter the order mxn:");
scanf("%d%d",&m,&n);
printf("Enter the matrix:\n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        scanf("%d",&a[i][j]);
    }
}
displayTranspose(a,m,n);
return 0;
}

```

```

void displayTranspose(int p[10][10],int m,int n)
{
    int i,j;
    printf("Transposed matrix is:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            printf("%d ",p[j][i]);
        }
        printf("\n");
    }
}

```

Sample Input/Output:

```

Enter the order mxn:3 2
Enter the matrix:
1 2
3 4
5 6
Transposed matrix is:
1 3 5
2 4 6

```

23) Write a program in C to calculate the product of two 3 x 3 matrices entered by the user using the concept of passing 2D arrays to a user defined function.

Answer:

```

#include<stdio.h>

void calcProduct(int [3][3],int[3][3],int [3][3]);

int main()
{
    int a[3][3],b[3][3],c[3][3],i,j;
    printf("Enter first matrix:\n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter second matrix:\n");
}

```

```

for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        scanf("%d",&b[i][j]);
    }
}
calcProduct(a,b,c);
printf("The product matrix is:\n");
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d ",c[i][j]);
    }
    printf("\n");
}
return 0;
}

void calcProduct(int a[3][3],int b[3][3],int c[3][3])
{
    int i,j,k,sum;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            sum=0;
            for(k=0;k<3;k++)
            {
                sum=sum+a[i][k]*b[k][j];
            }
            c[i][j]=sum;
        }
    }
}

```

Sample Input/Output:

```

Enter first matrix:
1 2 3
4 5 6
7 8 9
Enter second matrix:
1 1 1
2 2 2
3 3 3
The product matrix is:
14 14 14
32 32 32
50 50 50

```

Programs on passing strings to user defined functions

24) Write a program in C to calculate the total no. of characters and words in a line of string entered by the user by using the concept of passing string to the user defined functions.

Answer:

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

```
int findLength(char []);
```

```
int findWords(char []);
```

```
int main()
```

```
{
```

```
int nchars,nwords;
```

```
char str[100];
```

```
printf("Enter a string:");
```

```
gets(str);
```

```
nchars=findLength(str);
```

```
nwords=findWords(str);
```

```
printf("No. of characters=%d\n",nchars);
```

```
printf("No. of words=%d",nwords);
```

```
return 0;
```

```
}
```

```
int findLength(char p[])
```

```
{
```

```
int i=0;
```

```
while(p[i]!='\0')
```

```
{
```

```
i++;
```

```
}
```

```
return i;
```



```

}

int findWords(char p[])
{
    int i=0,count=1;
    while(p[i]!='\0')
    {
        if(p[i]==' ')
            count++;
        i++;
    }
    return count;
}

```

Sample Input/Output:

```

Enter a string:The quick brown fox jumps over the lazy dog
No. of characters=43
No. of words=9

```

Programs on Recursive Functions

25) Write a program in C to calculate the factorial of a number entered by the user using the concept of a recursive function.

Answer:

```

#include<stdio.h>

int fact(int);

int main()
{
    int n,f;
    printf("Enter a no:");
    scanf("%d",&n);
    f=fact(n);
    printf("The factorial is %d",f);
    return 0;
}

int fact(int n)
{
    if(n==0 || n==1)
        return 1;
    else
        return n*fact(n-1);
}

```

Sample Input/Output:

```
Enter a no:5
The factorial is 120
```

26) Write a program in C to calculate the sum of first n natural numbers using the concept of recursive functions.

Answer:

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

```
int sumnterms(int) ;
```

```
int main()
```

```
{
int n,s;
printf("Enter a no:");
scanf("%d",&n);
s=sumnterms(n);
printf("The is %d",s);
return 0;
}
```

```
int sumnterms(int n)
```

```
{
if(n==0)
    return 0;
else
    return n+sumnterms(n-1);
}
```

Sample Input/Output:

```
Enter n:10
The is 55
```

27) WAP in C to calculate the sum of squares of first n natural numbers using the concept of recursive function.

Answer:

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

```
int sumn2terms(int) ;
```

```
int main()
```

```
{
int n,s;
printf("Enter n:");
```

```

scanf ("%d", &n) ;
s=sumn2terms(n) ;
printf("The is %d",s) ;
return 0;
}

int sumn2terms(int n)
{
if(n==0)
    return 0;
else
    return n*n+sumn2terms(n-1) ;
}

```

Sample Input/Output:

```

Enter n:3
The is 14

```

28) WAP in C to calculate the sum of following series upto n terms.

$$S_n = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \text{upto } n \text{ terms}$$

Answer:

```

#include<stdio.h>

float summer(int);

int main()
{
int n;
float s;
printf("Enter n:");
scanf ("%d", &n) ;
s=summer(n) ;
printf("The required sum is %f",s) ;
return 0;
}

float summer(int n)
{
if(n==1)
    return 1;
else
    return pow(-1,n+1)*1.0/(2*n-1)+summer(n-1) ;
}

```

Sample Input/Output:

```
Enter n:3
The required sum is 0.866667
```

29) WAP to find the value of

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! + \dots \dots \dots \text{upto } n \text{ terms}$$

using recursive functions

Answer:

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

```
#include<math.h>
```

```
int fact (int);
```

```
float summer(float , int);
```

```
int main()
```

```
{
```

```
    float x,sum;
```

```
    int n;
```

```
    printf("enter the value of x:");
```

```
    scanf("%f",&x);
```

```
    printf("enter non-zero number of terms:");
```

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

```
    sum=summer(x,n);
```

```
    printf("the value of sin (%f) is:%f",x, sum);
```

```
    return 0;
```

```
}
```

```
int fact(int n)
```

```
{
```

```
    if(n==0||n==1)
```

```
        return 1;
```

```
    else
```

```
        return n*fact(n-1);
```

```
}
```

```
float summer(float x , int n )
```

```
{
```

```
if ( x==0 || n==0)
```

```
    return x;
```

```
else
```

```
    return pow(-1,n)*pow(x,2*n+1)/fact(2*n+1)+summer(x,n-1);
```

```
}
```

Sample Input/Output:

```
enter the value of x:1.5707
enter non-zero number of terms:15
the value of sin (1.570700) is:0.997785
```

30) Write a program in C to illustrate the concept of nested function.

Answer:

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

```
int main()
{
    auto void display();
    display();
    printf("Inside the main function\n");
    void display()
    {
        printf("Inside the display function\n");
    }
    return 0;
}
```

Sample Input/Output:

```
Inside the display function
Inside the main function
```

STRUCTURE THEORY

Structure is a logical collection of dissimilar types of data or variables. Use of structure in programs makes the programs more readable and efficient. Structure is usually defined to represent a record.

The syntax for **defining** a structure is as follows:-

```
struct structure_name
{
type variable_name1;
type variable_name1;
type variable_name1;
type variable_name1;
};
```

Example:

```
struct student
{
char name[20];
int roll;
int marks;
char address[30];
};
```

The above example shows how a structure definition can be written. The above code defines a new structure with name student embedding different variable like name, roll, marks and address.

After a structure has been defined, it can be **declared** in any function as follows:

```
struct structure_name variable_name;
```

Example:

```
struct student s;
```

The above statement declares a structure variable s of type student. The **struct** is the keyword.

The structure variable can also be declared along with the definition as follows:-

```
struct student
{
char name[20];
int roll;
int marks;
char address[30];
}s;
```

It defines the structure student as well as declares a structure variable named s of type student.

Initializing a structure:-

Suppose we have a structure named student already defined. The structure can be initialized like an array and string.

The statement,

```
struct student s = {"Ram", 12, 98, "Kathmandu"};
```

initializes s.name to "Ram", s.roll to 12, s.marks to 98 and s.address to "Kathmandu".

Accessing Structure Elements:-

Each element of structure can be accessed by using dot (.) operator. If we have defined a structure called student and have declared it as follows:

```
struct student s;
```

Each element of s can be accessed by using dot operator as follows:

s.name, s.roll, s.address etc.

Each of these can be treated as a separate variable. We can input values into them using functions like scanf() and display their values using functions like printf() as follows:

```
scanf("%s", s.name);  
printf("%s", s.name);
```

Array of Structure:-

One can create an array of structure like we created an array of basic types.

For example:

```
struct student s[48];
```

The above statement declares/creates an array of structure called student with size 48. This means 48 structures ranging from s[0], s[1],... up to s[47] has been created at once.

The member can be accessed as s[0].name, s[0].roll, s[0].address, s[1].name, s[1].roll, s[1].address, etc.

PROGRAMS ON STRUCTURE

1. WAP in C to input the details of a student from the user and display the entered details using the concept of structure

Answer:

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

```
struct student
```

```
{
```

```
int roll;
```

```
char name[20];
```

```
float marks;
```

```
char address[40];
```

```
};
```

```
int main()
```

```
{
```

```
struct student s;
```

```
printf("Enter the name,roll,marks and address:\n");
```

```
scanf("%s%d%f%s",s.name,&s.roll,&s.marks,s.address);
```

```
printf("The entered details of student is:\n");
```

```
printf("%s\t%d\t%f\t%s",s.name,s.roll,s.marks,s.address);
```

```
return 0;
```

```
}
```

Sample Input/Output:

```
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
The entered details of student is:
Ram      12      98.500000      Palpa
Process returned 0 (0x0)   execution time = 0.000000 sec
Press any key to continue.
```

2. WAP in C to illustrate the concept of initializing the members of a structure variable and display the initialized details

Answer:

```
#include<stdio.h>

struct student
{
int roll;
char name[20];
float marks;
char address[40];
};

int main()
{
struct student s={12,"Ram",98.5,"Palpa"};
printf("The entered details of student is:\n");
printf("%s\t%d\t%f\t%s",s.name,s.roll,s.marks,s.address);
return 0;
}
```

Sample Output:

```
The initialized details of student is:
Ram      12      98.500000      Palpa
```

3. WAP in C to input the details of n students from the user and display the entered details using the concept of an array of structure

Answer:

```
#include<stdio.h>

struct student
{
int roll;
char name[20];
float marks;
char address[40];
}
```



```

};

int main()
{
    struct student s[48];
    int n,i;
    printf("Enter no. of students:\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the name,roll,marks and address:\n");
        scanf("%s%d%f%s",s[i].name,&s[i].roll,&s[i].marks,s[i].address);
    }
    printf("The entered details of student is:\n");
    printf("Name\tRoll\tMarks\tAddress\n");
    for(i=0;i<n;i++)
    {
        printf("%s\t%d\t%f\t%s",s[i].name,s[i].roll,s[i].marks,s[i].address);
    }
    return 0;
}

```

Sample Input/Output:

```

Enter no. of students:
2
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
Enter the name,roll,marks and address:
Hari 13 99 Tansen
The entered details of student is:
Name      Roll      Marks      Address
Ram       12       98.500000   Palpa
Hari      13       99.000000   Tansen

```

4. WAP in C to input the details of n students from the user and display the details of only those students whose marks is greater than or equal to 80.

Answer:

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

```

struct student
{
    int roll;
    char name[20];
    float marks;
    char address[40];
}

```

```

};

int main()
{
    struct student s[48];
    int n,i;
    printf("Enter no. of students:\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the name,roll,marks and address:\n");
        scanf("%s%d%f%s",s[i].name,&s[i].roll,&s[i].marks,s[i].address);
    }
    printf("The entered details of student is:\n");
    printf("Name\tRoll\tMarks\tAddress\n");
    for(i=0;i<n;i++)
    {
        if(s[i].marks>=80)
            printf("%s\t%d\t%f\t%s",s[i].name,s[i].roll,s[i].marks,s[i].address);
    }
    return 0;
}

```

Sample Input/Output:

```

Enter no. of students:
3
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
Enter the name,roll,marks and address:
Hari 13 70 Tansen
Enter the name,roll,marks and address:
Shyam 14 80 KTM
The entered details of student is:
Name      Roll      Marks      Address
Ram        12         98.500000  Palpa
Shyam      14         80.000000  KTM

```

5. WAP in C to input the details of n students from the user and display the entered details in the order of marks.

Answer:

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

```
struct student
```

```

{
int roll;
char name[20];
float marks;
char address[40];
};

int main()
{
struct student s[48],temp;
int n,i,j;
printf("Enter no. of students:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the name,roll,marks and address:\n");
scanf("%s%d%f%s",s[i].name,&s[i].roll,&s[i].marks,s[i].address
);
}
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(s[i].marks<s[j].marks)
{
temp=s[i];
s[i]=s[j];
s[j]=temp;
}
}
}
printf("The entered details in order of marks:\n");
printf("Name\tRoll\tMarks\tAddress\n");
for(i=0;i<n;i++)
{
printf("%s\t%d\t%f\t%s\n",s[i].name,s[i].roll,s[i].marks,s[i].
address);
}
return 0;
}

```

Sample Input/Output:

```
Enter no. of students:
5
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
Enter the name,roll,marks and address:
Hari 13 99 Tansen
Enter the name,roll,marks and address:
Shyam 14 99.5 KTM
Enter the name,roll,marks and address:
Gita 11 78 PKR
Enter the name,roll,marks and address:
Rita 8 79 TNU
The entered details in order of marks:
Name      Roll      Marks      Address
Shyam     14       99.500000      KTM
Hari      13       99.000000      Tansen
Ram       12       98.500000      Palpa
Rita      8       79.000000      TNU
Gita     11       78.000000      PKR
```

6. WAP in C to input the details of a student from the user and display the entered details using the concept of passing structure to function

Answer:

```
#include<stdio.h>

struct student
{
int roll;
char name[20];
float marks;
char address[40];
};

void display(struct student);

int main()
{
struct student s;
printf("Enter the name,roll,marks and address:\n");
scanf("%s%d%f%s",s.name,&s.roll,&s.marks,s.address);
display(s);
return 0;
}
```

```

void display(struct student p)
{
printf("The entered details of student is:\n");
printf("%s\t%d\t%f\t%s",p.name,p.roll,p.marks,p.address);
}

```

Sample Input/Output:

```

Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
The entered details of student is:
Ram      12      98.500000      Palpa

```

7. WAP in C to input the details of n students from the user and display the entered details using the concept of passing array of structure to function

Answer:

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

```

struct student
{
int roll;
char name[20];
float marks;
char address[40];
};

```

```
void display(struct student [],int);
```

```

int main()
{
struct student s[48];
int n,i;
printf("Enter no. of students:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the name,roll,marks and address:\n");
scanf("%s%d%f%s",s[i].name,&s[i].roll,&s[i].marks,s[i].address);
}
display(s,n);
return 0;
}

```

```

void display(struct student p[],int n)
{
printf("The entered details of student is:\n");
printf("Name\tRoll\tMarks\tAddress\n");
for(i=0;i<n;i++)
{
printf("%s\t%d\t%f\t%s",p[i].name,p[i].roll,p[i].marks,p[i].address);
}
}

```

Sample Input/Output:

```

Enter no. of students:
3
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
Enter the name,roll,marks and address:
Hari 13 99 Tansen
Enter the name,roll,marks and address:
Shyam 14 99.5 PKR
The entered details of student are:
Ram      12      98.500000      Palpa
Hari     13      99.000000      Tansen
Shyam    14      99.500000      PKR

```

8. WAP in C to illustrate the concept of typedef in structure

Answer:

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

```
typedef struct
```

```
{
int roll;
char name[20];
float marks;
char address[40];
}student;
```

```
int main()
```

```
{
student s;
printf("Enter the name,roll,marks and address:\n");
scanf("%s%d%f%s",s.name,&s.roll,&s.marks,s.address);
printf("The entered details of student is:\n");
printf("%s\t%d\t%f\t%s",s.name,s.roll,s.marks,s.address);
}
```

```
return 0;
}
```

Sample Input/Output:

```
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
The entered details of student is:
Ram      12      98.500000      Palpa
Process returned 0 (0x0)   execution time 0.000000 sec
Press any key to continue.
```

9. WAP in C to illustrate the concept of nested structure

Answer:

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

```
struct DOB
{
int year;
int month;
int day;
};
```

```
struct person
{
char name[20];
int ctzn;
struct DOB d;
};
```

```
int main()
{
struct person p;
printf("Enter the name and ctzn:\n");
scanf("%s%d",p.name,&p.ctzn);
printf("Enter DOB:YYYY-MM-DD:");
scanf("%d%d%d",&p.d.year,&p.d.month,&p.d.day);
printf("The entered details of person is:\n");
printf("Name=%s, CTZN=%d\n",p.name,p.ctzn);
printf("DOB:%d-%d-%d",p.d.year,p.d.month,p.d.day);
return 0;
}
```

Sample Input/Output:

```
Enter the name and ctzn:
Ram 12345
Enter DOB:YYYY-MM-DD:2055
12 12
The entered details of person is:
Name=Ram, CTZN=12345
DOB:2055-12-12
```

Union Theory:

Union is similar to structure with some differences. First, all member of structure have different addresses and sizes. Secondly, the total size of the structure is equal to the sum of the size of each member. For example: If we have the structure student defined as:

```
struct student
{
int roll;
char name[20];
float marks;
char address[50];
}
```

Assuming the size of integer is 2 bytes, character is 1 byte and float is 4 bytes, the size of member variables roll, name, marks and address are 2 bytes, 20 bytes, 4 bytes and 50 bytes respectively and the total size of structure student is 76 bytes.

In case of union, all member of structure share the common address. And the total size of the union is equal to the size of largest member contained in it. While using union, since all members of the union share common memory address, only one member can be accessed at a time.

The syntax for defining a union is:

```
union union_name
{
type1 variable_name1;
type2 variable_name2;
type3 variable_name3;
type4 variable_name4;
}
```

Once the union has been defined a variable of type union can be declared using following syntax:

```
union union_name variable_name;
For example:
union student
{
int roll;
char name[20];
```



```
float marks;
char address[50];
}
```

In the above union definition, the size of largest member is 50 bytes. And all other members share this memory. So the total size of union student is 50 bytes.

The statement,

```
union student s;
```

declares a union variable s of type student.

10 .Write a program in CPP to display the details of a student entered by the user using the concept of union.

Answer:

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

```
union student
{
int roll;
char name[20];
float marks;
char address[40];
};
```

```
int main()
{
union student s;
printf("Enter the name,roll,marks and address:\n");
scanf("%s%d%f%s",s.name,&s.roll,&s.marks,s.address);
printf("The entered details of student is:\n");
printf("%s\t%d\t%f\t%s",s.name,s.roll,s.marks,s.address);
return 0;
}
```

Sample Input/Output:

```
Enter the name,roll,marks and address:
Ram 12 98 Palpa
The entered details of student is:
Palpa 1886150992 292624423646634740000000000000.000000 Palpa
```

It is to be noted that the program has displayed the information incorrectly. It is due to the fact that only one member of the union can be accessed at a time.

Following code will fix the problem:

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

```

union student
{
int roll;
char name[20];
float marks;
char address[40];
};

int main()
{
union student s;
printf("Enter name:");
scanf("%s",s.name);
printf("Entered name is %s\n",s.name);
printf("Enter roll:");
scanf("%d",&s.roll);
printf("Entered roll is %d\n",s.roll);
printf("Enter marks:");
scanf("%f",&s.marks);
printf("Entered marks is %f\n",s.marks);
printf("Enter address:");
scanf("%s",s.address);
printf("Entered address is %s",s.address);
return 0;
}

```

Sample Input/Output:

```

Enter name:Ram
Entered name is Ram
Enter roll:12
Entered roll is 12
Enter marks:98
Entered marks is 98.000000
Enter address:Palpa
Entered address is Palpa

```

Pointer and Structure:

Assuming that a structure named student has been defined, the statement,

```
struct student s;
```

declares a structure variable s of type student.

The structure variable s has four members, s.name, s.roll, s.marks and s.address

If we declare a pointer variable ptr of type student and if memory address of s is assigned to ptr, then ptr becomes pointer to s.

```
struct student *ptr;
ptr = &s;
```

In such situation, we can use pointer variable ptr to access the members of s.

ptr -> name can be used to access **s.name**

ptr -> roll can be used to access **s.roll**

ptr -> marks can be used to access **s.marks**

ptr -> address can be used to access **s.address**

11.WAP in C to input the details of a student from the user and display the entered details using the concept of structure and pointer

Answer:

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

```
struct student
```

```
{
```

```
int roll;
```

```
char name[20];
```

```
float marks;
```

```
char address[40];
```

```
};
```

```
int main()
```

```
{
```

```
struct student s,*ptr;
```

```
ptr=&s;
```

```
printf("Enter the name,roll,marks and address:\n");
```

```
scanf("%s%d%f%s",ptr->name,&ptr->roll,&ptr->marks,ptr->address);
```

```
printf("The entered details of student is:\n");
```

```
printf("%s\t%d\t%f\t%s",ptr->name,ptr->roll,ptr->marks,ptr->address);
```

```
return 0;
```

```
}
```

Sample Input/Output:

```
Enter the name,roll,marks and address:
Ram 12 98 Palpa
The entered details of student is:
Ram      12      98.000000      Palpa
```

12. Write a program in C to input the details of n students from the user and display the entered details using the concept of pointer and structure array.

Answer:

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

```
struct student
```

```
{
```

```

int roll;
char name[20];
float marks;
char address[40];
};

int main()
{
    struct student s[48],*ptr;
    int n,i;
    ptr=&s[0];
    printf("Enter no. of students:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the name,roll,marks and address:\n");
        scanf("%s%d%f%s", (ptr+i)->name, &(ptr+i)->roll, &(ptr+i)->marks, (ptr+i)->address);
    }
    printf("The entered details of student is:\n");
    printf("Name\tRoll\tMarks\tAddress\n");
    for(i=0;i<n;i++)
    {
        printf("%s\t%d\t%f\t%s\n", (ptr+i)->name, (ptr+i)->roll, (ptr+i)->marks, (ptr+i)->address);
    }
    return 0;
}

```

Sample Input/Output:

```

Enter no. of students:2
Enter the name,roll,marks and address:
Ram 12 98 Palpa
Enter the name,roll,marks and address:
Hari 13 99 Tansen
The entered details of student is:
Name      Roll      Marks      Address
Ram       12        98.000000   Palpa
Hari      13        99.000000   Tansen

```

PROGRAMS ON POINTERS

BASIC POINTER CONCEPT

1. WAP in C to illustrate the concept of pointer

```
#include<stdio.h>

int main()
{
    int a,*ptr;
    ptr=&a;
    printf("Enter the value of a:");
    scanf("%d",&a);
    printf("Value of a=%d\n",a);
    printf("Memory address of a=%d\n",&a);
    printf("Value of a=%d\n",*ptr);
    printf("Memory address of a=%d\n",ptr);
    printf("Memory address of ptr=%d",&ptr);
    return 0;
}
```

Sample Input/output:

```
Enter the value of a:5
Value of a=5
Memory address of a=2686748
Value of a=5
Memory address of a=2686748
Memory address of ptr=2686744
```

2. WAP in C to add two integers entered by the user using the concept of pointer

Answer:

```
#include<stdio.h>

int main()
{
    int a,b,*aptr=&a,*bptr=&b;
    printf("Enter two numbers:");
    scanf("%d%d",aptr,bptr);
    printf("The sum of %d and %d is %d",*aptr,*bptr,*aptr+*bptr);
    return 0;
}
```

Sample Input/Output:

```
Enter two numbers:2 3
The sum of 2 and 3 is 5
```

POINTER ARITHMETIC

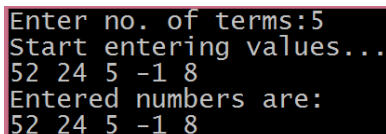
3. WAP in C to input n numbers from the user and display the entered numbers using the concept of pointer arithmetic

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],*ptr,n,i;
    ptr=&a[0];
    printf("Enter no. of terms:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d", (ptr+i));
    }
    printf("Entered numbers are:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",*(ptr+i));
    }
    return 0;
}
```

Sample Input/output:



```
Enter no. of terms:5
Start entering values...
52 24 5 -1 8
Entered numbers are:
52 24 5 -1 8
```

RELATION BETWEEN POINTER AND ARRAY

4. WAP in C to input n numbers from the user and display the entered numbers using the concept of relationship between pointer and array

Answer:

```
#include<stdio.h>

int main()
{
    int a[100],n,i;
    printf("Enter no. of terms:");
    scanf("%d",&n);
    printf("Start entering values...\n");
    for(i=0;i<n;i++)
    {
        scanf("%d", (a+i));
    }
}
```

```

}
printf("Entered numbers are:\n");
for(i=0;i<n;i++)
{
printf("%d ",*(a+i));
}
return 0;
}

```

Sample Input/output:

```

Enter no. of terms:5
Start entering values...
52 24 5 -1 8
Entered numbers are:
52 24 5 -1 8

```

5. WAP in C to sort n numbers entered by the user in ascending order using the concept of pointers.

```

#include<stdio.h>

int main()
{
int a[100],n,i,j,temp,*ptr;
ptr=&a[0];
printf("Enter no. of terms:");
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%d", (ptr+i));
}
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(*(ptr+i)>*(ptr+j))
{
temp=*(ptr+i);
*(ptr+i)=*(ptr+j);
*(ptr+j)=temp;
}
}
}
printf("In ascending order:\n");
for(i=0;i<n;i++)
{
printf("%d ",*(ptr+i));
}
return 0;
}

```

```
}
```

Sample Input/output:

```
Enter no. of terms:5
Start entering values...
52 24 5 -1 8
In ascending order:
-1 5 8 24 52
```

PROGRAMS USING THE CONCEPT OF PASSING POINTER TO A USER DEFINED FUNCTION

6. WAP in C to add two integers entered by the user using the concept of passing pointers to a user defined function

```
#include<stdio.h>

int addition(int *,int *);

int main()
{
    int a,b,sum;
    int *aptr=&a,*bptr=&b;
    printf("Enter two numbers:");
    scanf("%d%d",aptr,bptr);
    sum=addition(aptr,bptr);
    printf("The sum of %d and %d is %d",*aptr,*bptr,sum);
    return 0;
}

int addition(int *p,int *q)
{
    return *p+*q;
}
```

Sample Input/Output:

```
Enter two numbers:2 3
The sum of 2 and 3 is 5
```

7. WAP in C to input n numbers from the user and display the entered numbers using the concept of passing pointer to the function

```
#include<stdio.h>

void display(int *,int);

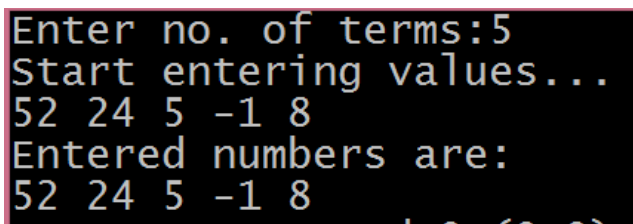
int main()
{
    int a[100],*ptr,n,i;
    ptr=&a[0];
    printf("Enter no. of terms:");
```



```
scanf("%d",&n);
printf("Start entering values...\n");
for(i=0;i<n;i++)
{
scanf("%d", (ptr+i));
}
display(ptr,n);
return 0;
}
```

```
void display(int *p, int n)
{
int i;
printf("Entered numbers are:\n");
for(i=0;i<n;i++)
{
printf("%d ",*(p+i));
}
}
```

Sample Input/Output



```
Enter no. of terms:5
Start entering values...
52 24 5 -1 8
Entered numbers are:
52 24 5 -1 8
```

POINTERS AND CHARACTER ARRAY

8. WAP in C to copy a string entered by the user using the concept of pointer and character array.

```
#include<stdio.h>

int main()
{
int i=0;
char str1[30],str2[30];
char *ptr1,*ptr2;
ptr1=&str1[0];
ptr2=&str2[0];
printf("Enter a string:");
scanf("%s",ptr1);
while(*(ptr1+i)!='\0')
{
*(ptr2+i)=*(ptr1+i);
i++;
}
printf("Copied string is %s",ptr2);
return 0;
}
```

```
}
```

Sample Input/Output:

```
Enter a string:Pokhara
Copied string is Pokhara
```

POINTER AND STRUCTURE

9. WAP in C to input the details of a student from the user and display the entered details using the concept of pointer to structure

Answer:

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

```
struct student
```

```
{
```

```
int roll;
```

```
char name[20];
```

```
float marks;
```

```
char address[40];
```

```
};
```

```
int main()
```

```
{
```

```
struct student s,*ptr;
```

```
ptr=&s;
```

```
printf("Enter the name,roll,marks and address:\n");
```

```
scanf("%s%d%f%s",ptr->name,&ptr->roll,&ptr->marks,ptr->address);
```

```
printf("The entered details of student is:\n");
```

```
printf("%s\t%d\t%f\t%s",ptr->name,ptr->roll,ptr->marks,ptr->address);
```

```
return 0;
```

```
}
```

Sample Input/Output:

```
Enter the name,roll,marks and address:
Ram 12 98.5 Palpa
The entered details of student is:
Ram      12      98.500000      Palpa
```

CHAIN OF POINTERS

10. Write the output of the following program

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

```
int main()
```

```
{
```

```
int a=5,*p,**q,***r,****s;
```

```
p=&a;
```

```

q=&p;
r=&q;
s=&r;
printf("Value of a=%d\n",a);
printf("Value of a=%d\n",*p);
printf("Value of a=%d\n",**q);
printf("Value of a=%d\n",***r);
printf("Value of a=%d",****s);
return 0;
}

```

Output:

```

Value of a=5
Value of a=5
Value of a=5
Value of a=5
Value of a=5

```

ARRAY OF POINTERS

11.WAP in C to illustrate the concept of array of pointers

Answer:

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

```

int main()
{
int i;
int a=24,b=63,c=-5,d=8,e=1;
int *ptr[5];
ptr[0]=&a;
ptr[1]=&b;
ptr[2]=&c;
ptr[3]=&d;
ptr[4]=&e;
for(i=0;i<5;i++)
{
printf("%d ",*ptr[i]);
}
return 0;
}

```

Output:

```
24 63 -5 8 1
```

FUNCTION RETURNING POINTERS

12.WAP in C to add two integers using the concept of returning pointer from a function

Answer:

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

```

int * addition(int a,int b);

int main()
{
int a,b,*sum;
printf("Enter two numbers:");
scanf("%d%d",&a,&b);
sum=addition(a,b);
printf("The sum %d and %d is %d",a,b,*sum);
return 0;
}

int * addition(int p,int q)
{
static int s;
s=p+q;
return &s;
}

```

Sample Input/Output:

```

Enter two numbers:2 3
The sum 2 and 3 is 5

```

13.WAP in C to input 5 numbers from the user and sort those entered numbers inside a user defined function. Return these numbers to the main function and display using the concept pointer.

Answer:

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

```

int * inputandSort();
void swap(int *p,int *q);

int main()
{
int i;
int *ptr;
ptr=inputandSort();
printf("In ascending order:\n");
for(i=0;i<5;i++)
{
printf("%d ",*(ptr+i));
}
return 0;
}

int * inputandSort()
{
static int a[5];

```

```

int i,j,temp;
printf("Enter five numbers:\n");
for(i=0;i<5;i++)
{
    scanf("%d",&a[i]);
}
for(i=0;i<4;i++)
{
    for(j=i+1;j<5;j++)
    {
        if(a[i]>a[j])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
}
return a;
}

```

Sample Input/Output:

```

Enter five numbers:
24 63 -5 8 1
In ascending order:
-5 1 8 24 63

```

DYNAMIC MEMORY ALLOCATION

In C programming language, it is required to specify the size of an array at compile time. But the user may not be able to do so always. The initial judgment of the size done by the user, if wrong, may cause failure of the program or wastage of the memory. For example: Suppose, the program is required to input integers until the user says NO and find the resulting sum. In the mentioned program if we had declared the integer array as:

int a[10];

and if user tries to input more than 10 integers, in such case array out of the bound error may occur.

Had we declared the integer array as:

int a[100];

and if user tries to input just 5 integers, then in such case memory space equivalent to that occupied by 95 integers would be wasted.

To solve the aforementioned problems, memory can be managed at the run time. The process of allocating memory at run time is known as dynamic memory allocation. DMA helps in efficient utilization of computer memory during program execution. In C programming language there are four memory management functions that can be used to allocate or free the memory during the program execution.

The four memory management functions and their task are summarized in the following table:

Table: *Memory Management Functions and their tasks*

Function	Task
malloc()	Allocates request size of bytes and returns a pointer to the first byte of the allocated space.
calloc()	Allocates space for an array of elements, initializes them to zero and then returns a pointer to the memory.
free()	Frees previously allocated space.
realloc()	Modifies the size of previously allocated space.

Memory allocation process

Figure shows the conceptual view of memory allocated for a C program.

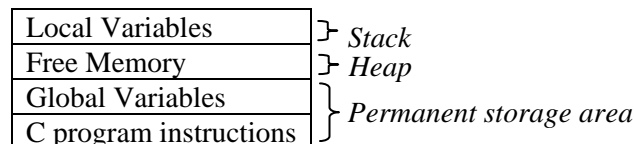


Fig: Memory allocation for a C program

The global variables and C program instructions (or statements) are stored in the permanent storage area. The local variables are stored in the storage area called stack. Stack is a region of memory where data is added or removed in a last-in-first-out (LIFO) manner. The memory region between stack and permanent storage area is available for use during run time. This free memory region is known as heap. Heap memory is also known as “dynamic memory”. The size of heap keeps changing when program is executed due to the creation and death of variable that are local to the functions or block. It is possible that heap memory might be full during the execution of the program. In such case the memory allocation function such as **malloc()** and **calloc()** might fail to allocate memory dynamically and return NULL instead indicating heap memory is full.

Different memory management functions are discussed below:

a. **malloc() function:**

malloc stands for memory allocation. **malloc()** function is used to allocate a block of memory dynamically. It reserves a block of memory of specified size and returns a pointer of type void. This means it can be type casted to any type of pointer. The syntax of **malloc()** is:

pointer_variable = (cast_type *) malloc(byte-size);

For example:

```
char *cptr;
cptr=(char *) malloc(10);
```

If above statements are executed, the **malloc()** function would allocate/reserve a memory space equivalent to 10 bytes. The memory address of the first byte is assigned to the pointer cptr of type char. However, if the heap is full and memory allocation fails, **malloc()** would return NULL.

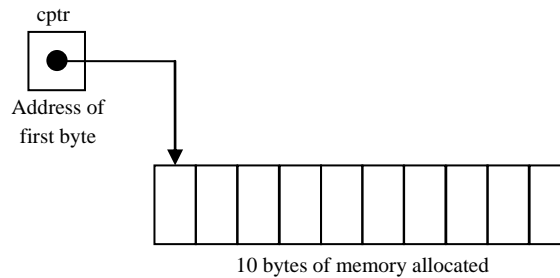


Fig: Dynamically allocated memory of 10 bytes

The storage space so allocated will have no name and can be accessed only through the use of pointer assigned.

Similarly, consider another example,

```
int *iptr;
iptr=(int*) malloc(50*sizeof(int));
```

If above statements are executed, the malloc() function would allocate/reserve a memory space equivalent to 50 times the size of an integer dynamically. The memory address of the first byte is assigned to the pointer iptr of type int. However, if the heap is full and memory allocation fails, malloc() would return NULL.

malloc() function can also be used to allocate memory dynamically for complex data type such as structures. Consider the following example:

```
struct student
{
int roll;
char name[20];
float marks;
char address[50];
};
struct student *sptr;
sptr = (struct student *) malloc(10*sizeof(struct student));
```

If above statements are executed, the malloc() function would allocate/reserve a memory space equivalent to 10 times the size of a structure of type student dynamically. The memory address of the first byte is assigned to the pointer sptr of type struct student. However, if the heap is full and memory allocation fails, malloc() would return NULL.

b. calloc() function:

calloc() function is used for allocating multiple blocks of storage dynamically, each of same size. calloc() function is generally used for allocating memory dynamically for data types such as arrays and structures. The syntax of calloc() function is:

pointer_variable = (cast_type *) calloc(n,size);

For example:

```
int *iptr;
iptr =(int *) calloc(15, sizeof(int));
```

If above statements are executed, the calloc() function would allocate contiguous memory space for 15 blocks dynamically. Each block is equivalent to the size of an integer. The memory address of the first byte of the allocated memory is assigned to the pointer iptr of type int. However, if the heap is full and memory allocation fails, calloc() would return NULL.

Consider another example,

```
struct student *sptr;  
sptr=(struct student *) calloc(10,sizeof(struct student));
```

If above statements are executed, the `calloc()` function would allocate contiguous memory space for 10 blocks dynamically. Each block is equivalent to the size of `struct student`. The memory address of the first byte of the allocated memory is assigned to the pointer `sptr` of type `struct student`. However, if the heap is full and memory allocation fails, `calloc()` would return `NULL`.

c. free() function:

The dynamically allocated memory can be released when the memory is not required. The release of memory is important when the storage space is limited. **free()** function can be used to clear the dynamically allocated memory. The syntax of **free()** function is:

free(pointer_variable);

For example:

free(iptr);

The above statement if executed, would free the memory space pointed by the pointer `iptr`. `Ip`tr might represent the pointer to the memory space allocated by **malloc()** or **calloc()** function.

d. realloc() function:

It is possible that the previously allocated memory might not be sufficient and we might need additional space for more elements. It is also possible that the previously allocated memory is much larger than necessary and we might want to reduce it. In both the cases, we can change the memory size already allocated with the help of function `realloc()`. This is called re-allocation of memory. The syntax of `realloc()` function is:

pointer_variable = realloc(pointer_variable, newsize);

For example:

```
int *iptr;  
iptr = (int *) malloc(50*sizeof(int));  
iptr=realloc(iptr,100*sizeof(int));
```

In the above example, `malloc()` function allocates memory space equivalent to the size of 50 integers. `realloc()` function then reallocates the memory to accommodate space for 100 integers.

Example programs on DMA:

1. Write a program in C to find the sum and average of 10 numbers using the concept of DMA with malloc function.

```
#include<stdio.h>  
#include<stdlib.h>  
  
int main()  
{  
int i;  
float *fptr,sum=0.00,avg;  
fptr=(float*)malloc(10*sizeof(float));  
printf("Enter 10 numbers:\n");  
for(i=0;i<10;i++)  
{
```



```

scanf("%f", (fptr+i));
sum=sum+*(fptr+i);
}
avg=sum/10;
printf("sum=%f and average=%f", sum, avg);
free(fptr);
return 0;
}

```

Sample Input/Output:

```

Enter 10 numbers:
1 2 3 4 5 6 7 8 9 10
sum=55.000000 and average=5.500000

```

2. Write a program in C to find the sum and average of 10 numbers using the concept of DMA with calloc function.

```

#include<stdio.h>
#include<stdlib.h>

int main()
{
    int i;
    float *fptr, sum=0.00, avg;
    fptr=(float*)calloc(10, sizeof(float));
    printf("Enter 10 numbers:\n");
    for(i=0; i<10; i++)
    {
        scanf("%f", (fptr+i));
        sum=sum+*(fptr+i);
    }
    avg=sum/10;
    printf("sum=%f and average=%f", sum, avg);
    free(fptr);
    return 0;
}

```

Sample Input/Output:

```

Enter 10 numbers:
1 2 3 4 5 6 7 8 9 10
sum=55.000000 and average=5.500000

```

3. Write a program in C to sort 5 integers entered by the user using the concept of DMA.

```

#include<stdio.h>
#include<stdlib.h>

int main()
{
    int i, j, *iptr, temp;
    iptr=(int*)malloc(10*sizeof(int));
    printf("Enter 5 numbers:\n");
}

```

```

for(i=0;i<5;i++)
{
scanf("%d", (iptr+i));
}
for(i=0;i<5;i++)
{
    for(j=i+1;j<5;j++)
    {
        if(*(iptr+i)>*(iptr+j))
        {
            temp=*(iptr+i);
            *(iptr+i)=*(iptr+j);
            *(iptr+j)=temp;
        }
    }
}
printf("Entered numbers in sorted order:\n");
for(i=0;i<5;i++)
{
printf("%d ",*(iptr+i));
}
free(iptr);
return 0;
}

```

Sample Input/Output

```

Enter 5 numbers:
24 63 -5 8 1
Entered numbers in sorted order:
-5 1 8 24 63

```

4. Write a program in C to input the name, roll and marks of six students and sort the entered information in the order of marks using the concept of DMA.

```

#include<stdio.h>
#include<stdlib.h>

```

```

struct student
{
char name[20];
int roll;
float marks;
};

```

```

int main()
{
int i,j;
struct student *sptr,temp;
sptr=(struct student *)malloc(6*sizeof(struct student));
printf("Enter the details:\n");
for(i=0;i<6;i++)

```

```

{
printf("Enter name, roll and marks:");
scanf("%s%d%f", (sptr+i)->name, &(sptr+i)->roll, &(sptr+i)->marks);
}
for(i=0; i<6; i++)
{
    for(j=i+1; j<6; j++)
    {
        if((sptr+i)->marks < (sptr+j)->marks)
        {
            temp = *(sptr+i);
            *(sptr+i) = *(sptr+j);
            *(sptr+j) = temp;
        }
    }
}
printf("In the order of marks\n");
printf("Name\tRoll\tMarks\n");
for(i=0; i<6; i++)
{
printf("%s\t%d\t%.2f\n", (sptr+i)->name, (sptr+i)->roll, (sptr+i)->marks);
}
free(sptr);
return 0;
}

```

Sample Input/Output:

```

Enter the details:
Enter name, roll and marks: Ram 12 98.5
Enter name, roll and marks: Hari 13 99
Enter name, roll and marks: Shyam 14 85
Enter name, roll and marks: Gita 15 89
Enter name, roll and marks: Rita 16 90
Enter name, roll and marks: Sita 17 67
In the order of marks
Name    Roll    Marks
Hari    13      99.00
Ram     12      98.50
Rita    16      90.00
Gita    15      89.00
Shyam   14      85.00
Sita    17      67.00

```

5. Write a program in C to illustrate the concept of re-allocation of memory. Your program should input integers from the user until the user says no and display the resulting sum.

```

#include<stdio.h>
#include<stdlib.h>

```

```

int main()
{
char ch;
int *iptr, count=0, sum=0;

```

```

iptr=(int *)malloc(10*sizeof(int));
do
{
count++;
if(count>=10)
    iptr=(int*)realloc(iptr,100*sizeof(int));
printf("Enter the integer:");
scanf("%d",iptr);
sum=sum+*iptr;
printf("Any more?(y/n):");
scanf(" %c",&ch);
}while(ch=='y' || ch=='Y');
printf("Total entered numbers=%d\n",count);
printf("Sum of entered numbers=%d",sum);
free(iptr);
return 0;
}

```

Sample Input/Output:

```

Enter the integer:5
Any more?(y/n):y
Enter the integer:6
Any more?(y/n):y
Enter the integer:7
Any more?(y/n):n
Total entered numbers=3
Sum of entered numbers=18

```

Unit 10

File Handling in C

Course Outline:-

-4 Hours

Concept of File, Opening and closing of File, Input Output Operations in File, Random Access in File, Error Handling in Files

Introduction:

A file is a collection of related information. File programming is different from Console I/O. In console I/O data can be read and displayed onto the screen but are not stored for future use whereas in file I/O, data can be saved onto files so that the information can be retrieved for the future usage.

File Operations:

In every file I/O, one has to do three basic operations. They are as follows:

- i. Opening a file.
- ii. Performing read, write, and append operation etc on an opened file.
- iii. Closing a file.

FILE Pointer:

FILE pointer is a pointer to structure of FILE type.

A file pointer can be declared as follows:

FILE *file_pointer;

Example:

FILE * fp;

In the above example statement, a file pointer variable fp has been declared. This file pointer fp can be used to perform file operations such as opening and closing files.

Opening a file:

The file pointer can be used to open the file as follows:

file_pointer = fopen("path_string_for_file", "mode_string");

Different modes of File Operations:-

Files can be opened in different modes depending upon the file operation(s) we wish to perform. Following are the different mode strings that define different modes of operation in the file IO:

w (write):

Open the file for writing purpose only.

r (read):

Open the file for reading purpose only.

a:

Open the file for appending or adding contents to it.

w+ (Write + Read):

Same as w except used for both reading and writing.

r+ (Read+Write):

The existing file is opened to beginning for both reading and writing.

a+ (append + read):

Same as a except both for reading and writing.

Example:

```
FILE *fp;
fp=fopen("student.txt","w");
/*The above statement opens file student.txt for writing purpose.*/
fp = fopen("student.txt", "r");
/*The above statement opens file student.txt for reading purpose.*/
fp = fopen("student.txt", "a");
/*The above statement opens file student.txt for appending purpose.*/
```

Closing a file:

A file can be closed by using `fclose()` function using the syntax as follows:

`fclose(file_pointer);`

where, `file_pointer` is a file pointer to opened file.

Example:

```
fclose(fp);
/*The above statement closes file pointed by fp, student.txt in our example.*/
```

Text File vs Binary Files:

On a computer, every file is a long string of ones and zeros. Specifically, a file is a finite-length sequence of bytes, where each byte is an integer between 0 and 255 inclusive (represented in binary as 00000000 to 11111111). Files can be broadly classified as either binary or text. These categories have different characteristics and need different tools to work with such files. Here is the primary difference between them:

Binary files have no inherent constraints (can be any sequence of bytes), and must be opened in an appropriate program that knows the specific file format (such as Media Player, Photoshop, Office, etc.). Text files must represent reasonable text and can be edited in *any* text editor program.

All files whether binary or text, are composed of bytes. The difference between binary and text files is in how these bytes are interpreted. Every text file is indeed a binary file, but this interpretation gives us no useful operations to work with. The reverse is not true, and treating a binary file as a text file can lead to data corruption.

While opening binary files, the mode strings are simply followed by `b` to that of text mode.

wb:

Open the binary file for writing purpose only.

rb:

Open the binary file for reading purpose only.

ab:

Open the binary file for appending or adding contents to it.

wb+:

Same as `wb` except used for both reading and writing.

rb+:

The existing binary file is opened to beginning for both reading and writing.

ab+:

Same as `ab` except both for reading and writing.

For Example:

```
FILE *fp;
fp=fopen("student.txt","wb");
/*The above statement opens a binary file student.txt for writing purpose.*/
fp = fopen("student.txt", "rb");
/*The above statement opens a binary file student.txt for reading purpose.*/
fp = fopen("student.txt", "ab");
/*The above statement opens a binary file student.txt for appending purpose.*/
```

END OF FILE (EOF)

EOF is a MACRO based integer value declared in stdio.h header file. When the end of the file is detected by the operating system, it automatically transmits EOF signal to indicate the end of the file. An attempt to read file after EOF signal might result in an error or infinite loop condition. The value of EOF in GNU GCC based compiler is -1. The hex value of EOF is 1A and decimal value is 26.

Character IO operation in Files:

In C programming language, character IO operation can be performed on a file using functions like fgetc() and fputc() using following syntax:

fputc(character_variable, file_pointer);

The function fputc() writes the value of character variable into the file pointed by the file_pointer.

character_variable = fgetc(file_pointer);

The function fgetc() reads a character from a file pointed by file_pointer and returns its value which can be assigned to another character variable.

1. WAP in C to write the characters entered by the user until the user press enter key

#include<stdio.h>

```
int main()
{
    char ch;
    FILE *fp;
    fp=fopen("student.txt","w");
    printf("Start writing to file...\n");
    while((ch=getchar())!='\n')
    {
        fputc(ch,fp);
    }
    printf("File Written!");
    fclose(fp);
    return 0;
}
```

Sample Input/Output:

```
Start writing to file...
The quick brown fox jumps over the lazy dog.
File Written!
```

2. WAP in C to read the file using the concept of character input from file

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

int main()
{
    char ch;
    FILE *fp;
    fp=fopen("student.txt","r");
    if(fp==NULL)
    {
        printf("Error reading file...\n");
        printf("Terminating...");
        exit(-1);
    }
    printf("Current contents of the file...\n");
    while((ch=fgetc(fp))!=EOF)
    {
        putchar(ch);
    }
    fclose(fp);
    return 0;
}
```

Sample Input/Output:

```
Current contents of the file...
The quick brown fox jumps over the lazy dog.
```

3. WAP in C to append the characters entered by the user in the existing file until the user press enter key

```
#include<stdio.h>

int main()
{
    char ch;
    FILE *fp;
    fp=fopen("student.txt","a");
    printf("Start appending to file...\n");
    while((ch=getchar())!='\n')
    {
        fputc(ch,fp);
    }
    printf("Contents Appended!");
    fclose(fp);
    return 0;
}
```


Sample Input/Output:

```
Start appending to file...
Nepal is a landlocked country.
Contents Appended!
```

4. WAP in C to copy the contents of one file into another

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

int main()
{
    char ch;
    FILE *sfp,*dfp;
    sfp=fopen("source.txt","r");
    dfp=fopen("destination.txt","w");
    if(sfp==NULL)
    {
        printf("Source file does not exist");
        printf("Terminating...");
        exit(-1);
    }
    while((ch=fgetc(sfp))!=EOF)
    {
        fputc(ch,dfp);
    }
    printf("File Copied!");
    fclose(sfp);
    fclose(dfp);
    return 0;
}
```

Sample Input/Output:

```
File Copied!
```

String IO operation in Files:

In C programming language, string IO operations can be performed on a file using functions like `fgets()` and `fputs()` using following syntax:

`fputs(string_variable,file_pointer);`

The function `fputs()` writes the contents of string variable into the file pointed by file pointer.

`fgets(string_variable,value,file_pointer);`

The function `fgets()` reads strings from the file pointed by `file_pointer` and copies it to the `string_variable`. The value represents an integer value which is the number of characters to be read from the file which is to be copied into the `string_variable`.

1. WAP in C to write strings entered by the user to the file until the user press Y or y.

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

```
int main()
{
FILE *fp;
fp=fopen("student.txt","w");
char str[100],ch;
do
{
printf("\nEnter the string:");
gets(str);
fputs(str,fp);
fputs("\n",fp);
puts("Any more?(y/n):");
ch=getche();
}while(ch=='y' || ch=='Y');
printf("\nFile written!");
fclose(fp);
return 0;
}
```

Sample Input/Output:

```
Enter the string:Nepal is a landlocked country
Any more?(y/n):
y
Enter the string:Mt. Everest is located in Nepal
Any more?(y/n):
n
File written!
```

2. WAP in C to read strings from the file written in previous question

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

```
#include<stdlib.h>
```

```
int main()
{
char str[100],ch;
FILE *fp;
```

```

fp=fopen("student.txt","r");
if(fp==NULL)
{
printf("Error opening file...");
exit(-1);
}
while(fgets(str,100,fp))
{
puts(str);
}
fclose(fp);
return 0;
}

```

Sample Input/Output:

Nepal is a landlocked countryMt. Everest is located in Nepal

Formatted IO operations in Files:

In C programming language, formatted IO operations can be performed on a file using functions like `fprintf()` and `fscanf()` using following syntax:

`fprintf(file_pointer, "control_string", list_of_arguments);`

The function `fprintf()` writes the value of arguments to the file being pointed by the `file_pointer` under the control of `control_string`.

`fscanf(file_pointer, "control_string",&list_of_arguments);`

The function `fscanf()` reads the formatted values from a file being pointed by the `file_pointer` into the list of arguments under the control of `control_string`.

1. WAP in C to illustrate the concept of formatted file output. Your program should input name, roll, marks and address of n students entered by the user and write it into a file

```

#include<stdio.h>
#include<stdlib.h>

```

```

int main()
{
int roll,n,i;
float marks;
char name[20],address[50];
FILE *fp;
fp=fopen("student.txt","w");
printf("Enter the number of students:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter name, roll, marks, and address:\n");
scanf("%s%d%f%s",name,&roll,&marks,address);
fprintf(fp,"%s %d %f %s\n",name,roll,marks,address);
}
}

```

```
printf("File Written!");
fclose(fp);
return 0;
}
```

Sample Input/Output:

```
Enter the number of students:3
Enter name, roll, marks, and address:
Ram 12 98.5 Palpa
Enter name, roll, marks, and address:
Hari 13 99 KTM
Enter name, roll, marks, and address:
Shyam 14 85 PKR
File Written!
```

2. WAP in C to illustrate the concept of formatted file input. Your program should read/input name, roll, marks and address of students within the file and display it in the console

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int roll;
float marks;
char name[20],address[50];
FILE *fp;
fp=fopen("student.txt","r");
if(fp==NULL)
{
printf("Error opening file...");
exit(-1);
}
printf("Current contents of the file:\n");
printf("Name\tRoll\tMarks\tAddress\n");
while(!feof(fp))
{
fscanf(fp,"%s%d%f%s",name,&roll,&marks,address);
printf("%s\t%d\t%.2f\t%s\t\n",name,roll,marks,address);
}
fclose(fp);
return 0;
}
```

Sample Input/Output:

```
Current contents of the file:
Name    Roll    Marks    Address
Ram     12      98.500000    Palpa
Hari    13      99.000000    KTM
Shyam   14      85.000000    PKR
```

Record IO operations in Files:

In C programming language, record IO operations can be performed on a file using functions like `fwrite()` and `fread()` using following syntax:

`fwrite(&structure_or_array_variable,sizeof_structure_or_array_variable,number_of_structure_or_array,file_pointer);`

The function `fwrite()` writes the contents of the structure or array into the file pointed by `file_pointer`.

`fread(&structure_or_array_variable,sizeof_structure_or_array_variable,number_of_structure_or_array,file_pointer);`

The function `fread()` reads the contents of the file pointed by the `file_pointer` and saves the read values into the structure variable.

1. WAP in C to write name, roll, marks and address of n students into a binary file "student.dat" using fwrite()

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

```
struct student
{
char name[20];
int roll;
float marks;
char address[40];
};

int main()
{
int n,i;
FILE *fp;
struct student s;
fp=fopen("student.dat","wb");
printf("Enter number of students:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter name,roll,marks and address:\n");
scanf("%s%d%f%s",s.name,&s.roll,&s.marks,s.address);
fwrite(&s,sizeof(struct student),1,fp);
}
printf("File Written!\n");
return 0;
}
```

Sample Input/Output:

```
Enter number of students:5
Enter name,roll,marks and address:
Ram 12 98.5 Palpa
Enter name,roll,marks and address:
Hari 13 99 KTM
Enter name,roll,marks and address:
Shyam 14 85 BKT
Enter name,roll,marks and address:
Rita 15 88 LAL
Enter name,roll,marks and address:
Sita 16 68 TNU
File Written!
```

2. WAP in C to read name, roll, marks and address of students from a binary file "student.dat" created in the previous example using fread()

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

struct student
{
    char name[20];
    int roll;
    float marks;
    char address[40];
};

int main()
{
    int i;
    FILE *fp;
    struct student s;
    fp=fopen("student.dat","rb");
    if(fp==NULL)
    {
        printf("Error opening file...\n");
        exit(-1);
    }
    printf("Contents of the file\n");
    printf("Name\tRoll\tMarks\tAddress\n");
    while(!feof(fp))
    {
        fread(&s,sizeof(struct student),1,fp);
        if(!feof(fp))
            printf("%s\t%d\t%f\t%s\n",s.name,s.roll,s.marks,s.address);
    }
    fclose(fp);
    return 0;
}
```

Sample Input/Output:

Contents of the file			
Name	Roll	Marks	Address
Ram	12	98.500000	Palpa
Hari	13	99.000000	KTM
Shyam	14	85.000000	BKT
Rita	15	88.000000	LAL
Sita	16	68.000000	TNU

3. *WAP in C to read name, roll, marks and address of students from a binary file "student.dat" created in the example 1 using fread() and display the details in the order of marks.*

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

struct student
{
    char name[20];
    int roll;
    float marks;
    char address[40];
};

int main()
{
    int i,j,n=-1;
    FILE *fp;
    struct student s[48],temp;
    fp=fopen("student.dat","rb");
    if(fp==NULL)
    {
        printf("Error opening file...\n");
        exit(-1);
    }
    while(!feof(fp))
    {
        fread(&s[++n],sizeof(struct student),1,fp);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(s[i].marks<s[j].marks)
            {
                temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
        }
    }
    for(i=0;i<n;i++)
    {
        printf("%s\t%d\t%f\t%s\n",s[i].name,s[i].roll,s[i].marks,s[i].address);
    }
}
```

```

        s[j]=temp;
    }
}
printf("Contents of the file in order of marks\n");
printf("Name\tRoll\tMarks\tAddress\n");
for(i=0;i<n;i++)
{
printf("%s\t%d\t%f\t%s\n",s[i].name,s[i].roll,s[i].marks,s[i].
address);
}
fclose(fp);
return 0;
}

```

Sample Input/Output:

```

Contents of the file in order of marks
Name    Roll    Marks    Address
Hari    13      99.000000    KTM
Ram     12      98.500000    Palpa
Rita    15      88.000000    LAL
Shyam   14      85.000000    BKT
Sita    16      68.000000    TNU

```


Complete input output operation on files using sequential access:

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

```
struct student
```

```
{
char name[20];
int roll;
int semester;
char address[30];
int marks;
};
```

```
void writeRecords()
```

```
{
FILE *fp;
struct student s;
int i,ns;
fp=fopen("student.dat","wb");
printf("Enter number of students to write:");
scanf("%d",&ns);
for(i=0;i<ns;i++)
{
printf("Enter name,roll,semester,address and marks:\n");
scanf("%s%d%d%s%d",s.name,&s.roll,&s.semester,s.address,&s.marks);
fwrite(&s,sizeof(struct student),1,fp);
}
printf("File Written!");
fclose(fp);
}
```

```
void readRecords()
```

```
{
FILE *fp;
struct student s;
fp=fopen("student.dat","rb");
printf("Name\tRoll\tSemester\tAddress\tMarks\n");
while(!feof(fp))
{
fread(&s,sizeof(struct student),1,fp);
if(!feof(fp))
printf("%s\t%d\t%d\t%s\t%d\n",s.name,s.roll,s.semester,s.address,s.marks);
}
}
```

```

fclose(fp);
}

void searchRecord()
{
FILE *fp;
struct student s;
int roll;
fp=fopen("student.dat","rb");
printf("Enter roll number of student to be searched:");
scanf("%d",&roll);
printf("Name\tRoll\tSemester\tAddress\tMarks\n");
while(!feof(fp))
{
fread(&s,sizeof(struct student),1,fp);
if(s.roll==roll)
printf("%s\t%d\t%d\t%s\t%d\n",s.name,s.roll,s.semester,s.address,s.marks);
}
fclose(fp);
}

void addRecord()
{
FILE *fp;
struct student s;
fp=fopen("student.dat","ab+");
printf("Enter information of student to be added:\n");
printf("Enter name,roll,semester,address and marks:\n");
scanf("%s%d%d%s%d",s.name,&s.roll,&s.semester,s.address,&s.marks);
fwrite(&s,sizeof(struct student),1,fp);
printf("Record Added!\n");
printf("Contents of the file after addition.\n");
fclose(fp);
}

void modifyRecord() {
FILE *fp;
struct student s;
int roll, found = 0;
fp = fopen("student.dat", "rb");
printf("Enter roll number of student to be modified: ");
scanf("%d", &roll);

```

```

FILE *temp_fp = fopen("temp.dat", "wb");
while (fread(&s, sizeof(struct student), 1, fp))
{
    if (s.roll == roll)
    {
        found = 1;
        printf("Enter new information:\n");
        printf("Enter name,roll,semester,address and marks:\n");
        scanf("%s%d%d%s%d",s.name,&s.roll,&s.semester,s.address,&s.marks);
    }
    fwrite(&s, sizeof(struct student), 1, temp_fp);
}
fclose(fp);
fclose(temp_fp);
remove("student.dat");
rename("temp.dat", "student.dat");
if (found)
    printf("Record Modified!\n");
else
    printf("Record not found!\n");
printf("Content of file after record modification:\n");
}

void deleteRecord()
{
    FILE *fp;
    struct student s;
    int roll, found = 0;
    fp = fopen("student.dat","rb");
    FILE *temp_fp = fopen("temp.dat","wb");
    printf("Enter roll number of student to be deleted: ");
    scanf("%d",&roll);
    while (fread(&s, sizeof(struct student),1,fp))
    {
        if(s.roll != roll)
            fwrite(&s, sizeof(struct student),1,temp_fp);
        else
            found = 1;
    }
    fclose(fp);
    fclose(temp_fp);
    remove("student.dat");
    rename("temp.dat", "student.dat");
}

```

```

if (found)
printf("Record Deleted!\n");
else
printf("Record not found!\n");
printf("Content of file after record deletion:\n");
}

int main()
{
int n;
printf("What do you want to do?\n");
printf("Enter 1 to write records into file:\n");
printf("Enter 2 to read records from file:\n");
printf("Enter 3 to search specific student from file:\n");
printf("Enter 4 to add a student record to file:\n");
printf("Enter 5 to modify a student record in file:\n");
printf("Enter 6 to delete a record of student from file:\n");
scanf("%d",&n);
switch(n)
{
case 1:
writeRecords();
break;
case 2:
readRecords();
break;
case 3:
searchRecord();
break;
case 4:
addRecord();
readRecords();
break;
case 5:
modifyRecord();
readRecords();
break;
case 6:
deleteRecord();
readRecords();
break;
default:
printf("Please enter the correct choice!\n");
}/*Switch Closed*/

```

}

Error handling during files input/output operations:

Errors might occur during the input/output operations on file. Typical error situations are:

- i. Trying to read beyond end-of-file mark.
- ii. Device overflow
- iii. Trying to use a file that has not been opened.
- iv. Trying to perform an operation on a file, when the file is opened for another type of operation.
- v. Opening a file with an invalid filename.
- vi. Attempting to write to a write-protected file.

There are two functions which can be used to inquire about the error status of the file. They are `feof()` and `ferror()`.

The function `feof()` takes file pointer as an argument and returns non-zero integer value if all of the data from the specified file has been read, and returns zero otherwise.

Suppose `fp` is a file pointer pointing to a certain file. Following code:

```
if(feof(fp))  
    printf("The end of the file has been reached.\n");
```

would display the message "The end of the file has been reached." on reaching the end of the file condition.

The function `ferror()` takes file pointer as an argument and returns a non-zero integer if an error has been detected up to that point during processing. It returns zero otherwise.

Again, let's suppose `fp` is a file pointer pointing to a certain file. Following code:

```
if(ferror(fp)!=0)  
    printf("An error has occurred\n");
```

would print the error message, if the reading is not successful.

Finally, we know that whenever a file is opened using `fopen` function, a file pointer is returned. If the file cannot be opened for some reason, then the function returns a NULL Pointer. This facility can be used to test whether a file has been opened or not.

For Example:

```
if(fp==NULL)  
    printf("Error opening the file.\n");
```

Random Access in Files:

So far we have discussed writing and reading data sequentially into and from the file. However it is possible to have a random access to the file using functions like `fseek()`, `ftell()`, `rewind()` etc.

ftell():

`ftell` takes file pointer and return a number of type long that corresponds to the current position. This function is useful in saving the current position of a file, which can be used later in the programs. It takes the following form:

```
n = ftell(fp);
```

`n` would give the relative offset(in bytes) of the current position. This means that `n` bytes have already been read(or written).

rewind():

Rewind takes a file pointer and resets the position to the start of the file. For example, the statement,

```
rewind(fp);  
n=ftell(fp);
```

would assign 0 to n because the file position has been set to the start of the file by rewind()

fseek():

fseek function is used to move the file position to a desired location within the file. It takes the following form:

```
fseek(file_pointer,offset,position);
```

file_pointer is the pointer to the file concerned, offset is a number or variable of the long, and position is an integer number. The offset specifies the number of positions (bytes) to be moved from the location specified by the position. The position can take one of the following three values:

Value	Meaning
0	Beginning of the file
1	Current position
2	End of the file

The offset may be positive, meaning moving forwards, or negative, meaning moving backwards.

Following examples illustrates the operation of the fseek function:

Statement	Meaning
fseek(fp,0L,0)	Go to the beginning(similar to rewind)
fseek(fp,0L,1)	Stay at the current position
fseek(fp,0L,2)	Go to the end of the file, past last character
fseek(fp,m,0)	Go to (m+1) th byte in the file.
fseek(fp,m,1)	Go forward by m bytes
fseek(fp,-m,1)	Go backward by m bytes from current position.
fseek(fp,-m,2)	Go backward by m bytes from the end.

When the operation is successful, fseek returns a zero. If we attempt to move the file pointer beyond the file boundaries, an error occurs and fseek returns -1. It is good practice to check whether an error has occurred or not, before proceeding further.

Q. Re-write complete file IO program using the concept of random access to files.

Reference:-

Programming in ANSI C by E Balagurusamy

UNIT 11

INTRODUCTION TO GRAPHICS

Graphics programming is used to draw different geometrical shape such as line, arc, circle, ellipse, rectangle, square etc. It can be used to create objects with different colors and patterns. Simple animations can also be made by using graphics related functions. In C programming language graphics.h header file provides several functions related to graphics.

Co-ordinate system of computer screen:

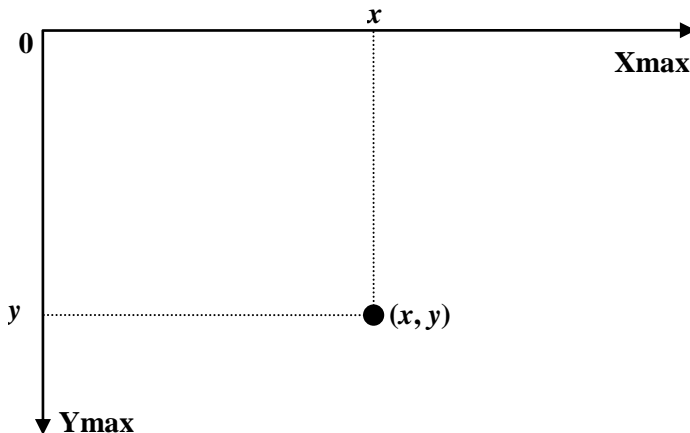


Figure: Co-ordinate system of a computer screen

The entire screen of a computer is divided into a matrix of pixels. For example: a 1024×576 resolution display has 589,824 pixels. Each pixel is identified by a co-ordinate system as shown in figure. The illumination of each pixel is controlled with a combination of red, blue and green light to form the required image. The top left portion of the display is identified as origin (0, 0) and bottom right portion of the display is identified as (Xmax, Ymax). The co-ordinates of a pixel is identified as (x, y).

Steps for Graphics Programming in C:

Following are the steps in graphics programming in C:

Step 1: Include the header file <graphics.h>

Step 2: Initialize the graphics driver using `intgraph()` function.

Step 3: Draw the required graph using functions in graphics.h

Step 4: Close the graph

Graphics function:

The functions which are used to output different objects onto a graphical window are known as graphics functions. Graphics functions helps to draw different geometrical objects such as line, circle, rectangle, ellipse etc. Graphics functions help to move to the specified pixel, output the text to the specified position, change the color of the background or the text/drawing object and many more. Different graphics functions available in C are summarized in the table below:

Table: *Graphics related functions in C Programming language*

Function	Purpose
line (x1, y1, x2, y2)	Draws a line from (x1,y1) to (x2,y2)
circle (xc, yc, r)	Draw a circle with center at (xc,yc) and radius r.
rectangle (x1, y1, x2, y2)	Draws a rectangle with top left corner at (x1,y1) and bottom right corner at (x2,y2).
ellipse (xc, yc, s, e, xr, yr)	Draws an ellipse with (xc,yc) as center. xr and yr are semi-major and semi-minor axes respectively.
arc (xc, yc, s, e, r)	Draws an arc with center (xc,yc), radius r, starting angle s and ending angle e.
gotoxy (x, y)	Moves the cursor at specified co-ordinate (x,y)
setcolor (color)	Sets the color of drawing object with given color. Color code might be any value in the range from 0 to 15.
outtextxy (x, y, "string") ;	Displays the string from (x,y) on screen
initgraph ()	Initializes the computer in graphics mode.
closegraph ()	Closes the graphics mode initialized by initgraph() function.
cleardevice ()	This function is used to clear the screen in graphical mode.

Note: In order to use above functions, user must include the header file <graphics.h> in the program.

Example Programs and Solved Past Questions

1) Write a program in C to output a text "Computer Programming" in red color at co-ordinate (100,100) on the screen.

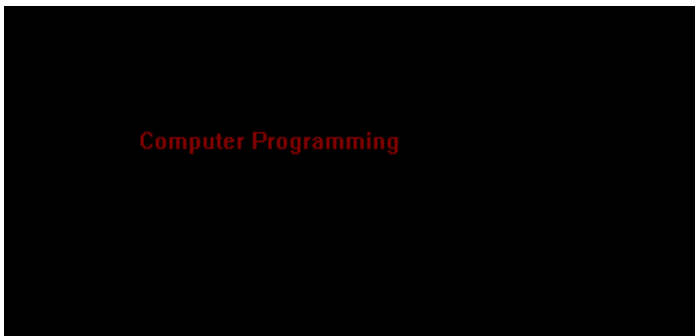
Answer:

```
#include<graphics.h>
#include<conio.h>

int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
setcolor(RED);
outtextxy(100,100,"Computer Programming");
getch();
closegraph();
return 0;
}
```

In the above program `initgraph()` function is used to initialize the graphics system. `gd` is used for graphics driver and `gm` is for graphics mode. The path " C:\\TC\\BGI" is the path to the graphics driver folder for TURBO C Compiler. In the above example, `gd` is set to `DETECT`, therefore, the resolution is set to highest value via `gm`.

Output:

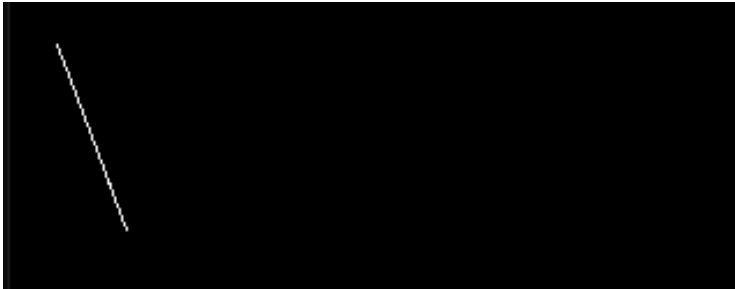


2) Write a program in C to draw a line from (20,20) to (50,100).

Answer:

```
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm," C:\\TC\\BGI ");
line(20,20,50,100);
getch();
closegraph();
return 0;
}
```

Output:



3) Write a program in C to draw a circle with center at (100,100) and radius 50 OR

Write a program to draw a circle.

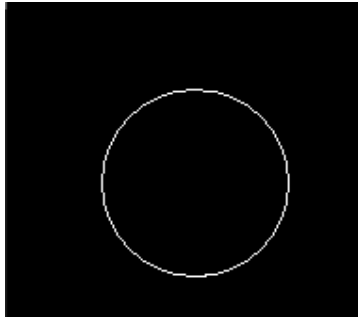
BSc CSIT 2077,2074,2073

Answer:

```
#include<graphics.h>
#include<conio.h>

int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm," C:\\TC\\BGI ");
circle(100,100,50);
getch();
closegraph();
return 0;
}
```

Output:



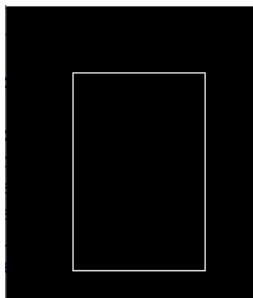
4) Write a program in C to draw a rectangle with top left corner (50,50) and bottom right corner (150,200)

Answer:

```
#include<graphics.h>
#include<conio.h>

int main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm," C:\\TC\\BGI ");
    rectangle(50,50,150,200);
    getch();
    closegraph();
    return 0;
}
```

Output:



- 5) Write a program in C to draw an arc with center at (100,100) with radius 50, starting angle =0 and ending angle =90.

Answer:

```
#include<graphics.h>
#include<conio.h>

int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm," C:\\TC\\BGI ");
arc(100,100,0,90,50);
getch();
closegraph();
return 0;
}
```

Output:



- 6) Write a program in C to draw an ellipse with center at (150,100) with xr=100 and yr= 40, starting angle =0 and ending angle =360.

Answer:

```
#include<graphics.h>
#include<conio.h>

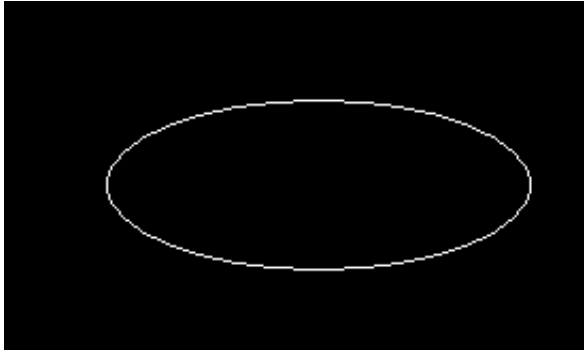
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
ellipse(150,100,0,360,100,40);
```

```

getch();
closegraph();
return 0;
}

```

Output:



7) Write short notes on graphics function. *B.Sc. CSIT 2075, 2071, 2069*

Answer:

The functions which are used to output different objects onto a graphical window are known as graphics functions. Graphics functions help to draw different geometrical objects such as line, circle, rectangle, ellipse etc. Graphics functions help to move to the specified pixel, output the text to the specified position, change the color of the background or the text/drawing object and many more. Different graphics functions available in C are summarized in the table below:

Table: *Graphics related functions in C Programming language*

Function	Purpose
line (x1, y1, x2, y2)	Draws a line from (x1,y1) to (x2,y2)
circle (xc, yc, r)	Draw a circle with center at (xc,yc) and radius r.
rectangle (x1, y1, x2, y2)	Draws a rectangle with top left corner at (x1,y1) and bottom right corner at (x2,y2).
ellipse (xc, yc, s, e, xr, yr)	Draws an ellipse with (xc,yc) as center. xr and yr are semi-major and semi-minor axes respectively.

arc(xc,yc,s,e,r)	Draws an arc with center (xc,yc), radius r, starting angle s and ending angle e.
gotoxy(x,y)	Moves the cursor at specified co-ordinate (x,y)
setcolor(color)	Sets the color of drawing object with given color. Color code might be any value in the range from 0 to 15.
outtextxy(x,y,"string") ;	Displays the string from (x,y) on screen
initgraph()	Initializes the computer in graphics mode.
closegraph()	Closes the graphics mode initialized by initgraph() function.
cleardevice()	This function is used to clear the screen in graphical mode.

In order to use graphics functions, the user must include the header file <graphics.h> in the program.

8) What is the use of initgraph() in C program? Write a program to draw a line, circle and rectangle using graphic function. BCA 2019

Answer: The main function/use of initgraph() function is to initialize the graphics system. For example:

```
int gd=DETECT, gm;
initgraph(&gd,&gm," C:\\TC\\BGI");
```

In the above example, gd is used for graphics driver and gm is for graphics mode. The path " C:\\TC\\BGI" is the path to the graphics driver folder for TURBO C Compiler. In the above example, gd is set to DETECT, therefore, the resolution is set to highest value via gm.

The required program is:

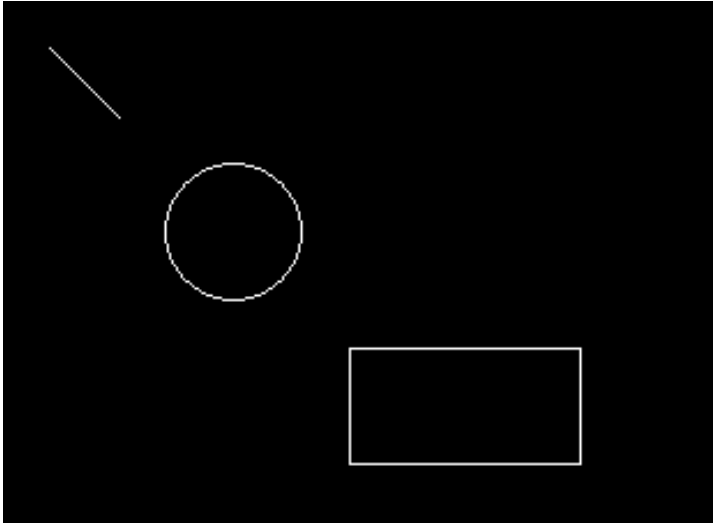
```
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
line(20,20,50,50);
```

```

circle(100,100,30);
rectangle(150,150,250,200);
getch();
closegraph();
return 0;
}

```

Output:



9) Explain any four graphics functions in C. Write a program to draw two concentric circles with center(50,50) and radii 75 and 125. BCA 2018

Answer: Any four graphics functions in C are given below:

i. line()

The syntax of line() function is:

```
line(x1,y1,x2,y2);
```

It draws a straight line from (x1,y1) to (x2,y2).

ii. circle()

The syntax of circle() function is:

```
circle(xc,yc,r);
```

It draws a circle with center at (xc,yc) and radius r.

iii. outtextxy()

The syntax of outtextxy() function is:

```
outtextxy(x,y,"string");
```

It display the "string" starting from co-ordinates (x,y).

iv. rectangle()

The syntax of rectangle() function is:

```
rectangle(x1,y1,x2,y2) ;
```

It draws a rectangle with top left corner (x1,y1) and bottom right corner (x2,y2).

For program part,

```
#include<graphics.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
int gd=DETECT,gm;
```

```
initgraph(&gd,&gm,"C:\\TC\\BGI");
```

```
circle(50,50,75);
```

```
circle(50,50,125);
```

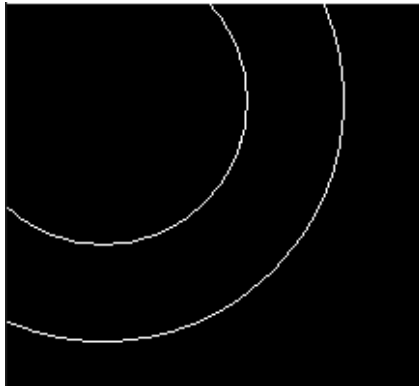
```
getch();
```

```
closegraph();
```

```
return 0;
```

```
}
```

Output:



In the output, we can observe some clipping. This is because the given radii for concentric circles are larger than the distance of their center from the x-axis and y axis.

10) Discuss with example any five graphics functions used in C Programming.

BSc CSIT 2072

Answer: Any five graphics functions in C are given below:

i. line()

The syntax of line() function is:

line(x1,y1,x2,y2);

It draws a straight line from (x1,y1) to (x2,y2).

ii. circle()

The syntax of circle() function is:

circle(xc,yc,r);

It draws a circle with center at (xc,yc) and radius r.

iii. outtextxy()

The syntax of outtextxy() function is:

outtextxy(x,y,"string");

It display the "string" starting from co-ordinates (x,y).

iv. rectangle()

The syntax of rectangle() function is:

rectangle(x1,y1,x2,y2);

It draws a rectangle with top left corner (x1,y1) and bottom right corner (x2,y2).

v. ellipse()

The syntax of ellipse() function is:

ellipse(xc,yc,s,e,xr,yr);

It draws an ellipse with (xc,yc) as center. xr and yr are semi-major and semi-minor axes respectively. s is starting angle and e is the ending angle.

Following example program illustrate the use of all of the above five functions:

```
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm," ");
outtextxy(100,10,"Graphics Programming");
line(20,20,50,50);
rectangle(20,20,100,80);
circle(50,50,40);
```

12

```
ellipse(100,100,0,360,40,50);  
getch();  
closegraph();  
return 0;  
}
```

Output:

