MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DSA LAB ASSIGNMENTS

SUBMITTED TO: MA’AM RITI KUSHWAHA

SUBMITTED BY: UJESH MAURYA

ID: 2015UCP1338

SEMESTER: 3 (2015-19)

SECTION: A3

Week 1

Problem 1: *Write a program to delete an element from an array, to be entered by user. Deletion position to be taken on runtime.*
Description: *The Programs aims at deletion of a particular element of a 0-based indexed array. It is useful in removing duplicate elements from an array. It takes an array of size 'n' and an index 'i' to be deleted, and prints the modified array with array[i] deleted.*
Solution:

```
#include<stdio.h>

int main()
{
      int a[1000],n,i,j,temp,p;
      printf("Enter the size of array\n");
      scanf("%d",&n); //Array size input
      printf("Enter the array\n");
      for(i=0;i<n;i++)
           scanf("%d",&a[i]);   //Array input
      printf("You have entered\n");
      for(i=0;i<n;i++)
           printf("%d ",a[i]);      //Array before deletion
      printf("\nEnter position from where element is to be deleted:\n");
      scanf("%d",&j);      //deletion position input
      for(i=j-1;i<n-1;i++)     //shifting the elements
           a[i]=a[i+1];
      printf("After insertion the array is:\n");
      for(i=0;i<n-1;i++)       //print modified array
           printf("%d ",a[i]);
}
```

Output:

```
Enter the size of array
5
Enter the array
1 2 3 4 5
You have entered
1 2 3 4 5
Enter position from where element is to be deleted:
2
After insertion the array is:
1 3 4 5
```

Problem 2: *Write a program to insert an element in an array to be entered by user. Insertion position to be taken at runtime.*

Description: *The Programs aims at insertion of a new element in an existing array. It proves to be useful in dynamically modifying an array or a string without really allocating a full new array. It takes an array of size 'n', an element 'm' and an index 'i' where 'm' is to be inserted, and prints the modified array with 'm' inserted at 'i' position.*

Solution:

```
#include<stdio.h>

int main()
{
	int a[1000],n,i,j,temp,p;
	printf("Enter the size of array\n");
	scanf("%d",&n);      //input size of array
	printf("Enter the array\n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);   //array elements input
	printf("You have entered\n");
	for(i=0;i<n;i++)
		printf("%d ",a[i]);      //print original array
	printf("\nEnter the element to be inserted:\n");
	scanf("%d",&p);      //input new element
	printf("Enter the position\n");
	scanf("%d",&j);      //input the position
	for(i=n-1;i>=j-1;i--)
		a[i+1]=a[i];     //shifting array
	a[j-1]=p;            //assigning new element
	printf("After insertion the array is:\n");
	for(i=0;i<n+1;i++)   //print modified array
		printf("%d ",a[i]);
}
```

Output:

```
Enter the size of array
5
Enter the array
1 2 3 4 5
You have entered
1 2 3 4 5
Enter the element to be inserted:
-1
```

```
Enter the position
2
After insertion the array is:
1 -1 2 3 4 5
```

____

Problem 3: *Write a program to calculate addition and multiplication of two matrices whose size and elements to be entered by user at runtime.*

Description: *The Programs aims at addition and multiplication of two matrices. It finds application in solving linear equations of n variables and n equations and various other purposes also. It takes two matrices 'A' and 'B' of size n*m and p*q respectively as input and prints the sum and multiplication of A and B matrix, if possible.*

Solution:

```
#include<stdio.h>
int main()
{
     int
n,m,i,j,k,p,q,a[100][100],b[100][100],c[100][100],d[100][100],flags=1,flagm=1;
     printf("Enter dimensions of matrix 1\n");
          scanf("%d %d",&n,&m);    //input n,m
     printf("Enter elements of matrix 1\n");
     for(i=0;i<n;i++)
          for(j=0;j<m;j++)
               scanf("%d",&a[i][j]);    //input A matrix
     printf("Enter dimensions of matrix 2\n");
          scanf("%d %d",&p,&q);    //input p,q
     printf("Enter elements of matrix 2\n");
     for(i=0;i<p;i++)
          for(j=0;j<q;j++)
               scanf("%d",&b[i][j]);    //input B matrix


     printf("1st Matrix:\n");
     for(i=0;i<n;i++)
     {
          for(j=0;j<m;j++)         //print A matrix
               printf("%d  ",a[i][j]);
          printf("\n");
     }
     printf("2nd Matrix\n");
     for(i=0;i<p;i++)
     {
          for(j=0;j<q;j++)         //print B matrix
               printf("%d  ",b[i][j]);
```

```
            printf("\n");
        }
        if((n!=p)&&(m!=q))
        {
            flags=0;        //condition for addition
            printf("Addition can’t be done\n");
        }
        if(flags==1)
        for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                c[i][j]=a[i][j]+b[i][j];     //perform addition
        for(i=0;i<n;i++)
            for(j=0;j<q;j++)
                d[i][j]=0;
        if(m!=p)        //condition for multiplication
        {
            flagm=0;
            printf("Multiplication can’t be done\n");
        }
    if(flagm==1)
        for(i=0;i<n;i++)
            for(j=0;j<q;j++)
            {
                for(k=0;k<m;k++)     //perform multiplication
                {
                    d[i][j]+=a[i][k]*b[k][j];
                }
            }
        if(flags==1)
        printf("SUM OF MATRIX :\n");
        if(flags==1)
        for(i=0;i<n;i++)     //print A+B matrix
        {
            for(j=0;j<m;j++)
                printf("%d  ",c[i][j]);
            printf("\n");
        }
        if(flagm==1)
        printf("PRODUCT OF MATRIX :\n");
        if(flagm==1)
        for(i=0;i<n;i++)
        {
            for(j=0;j<q;j++)     //print A*B matrix
```

```
                printf("%d  ",d[i][j]);
            printf("\n");
        }


}
```

Output:

```
1 2 3 4 5 6 7 8 9
Enter dimensions of matrix 2
3 3
Enter elements of matrix 2
-1 -2 -3 -4 -5 -6 -7 -8 -9
1st Matrix:
1  2  3
4  5  6
7  8  9
2nd Matrix
-1  -2  -3
-4  -5  -6
-7  -8  -9
SUM OF MATRIX :
0  0  0
0  0  0
0  0  0
PRODUCT OF MATRIX :
-30  -36  -42
-66  -81  -96
-102  -126  -150
```

____

Problem 4: *Write a program to add two polynomials whose degrees and coefficients are to be entered by user at runtime.*

Description: *The Programs aims at addition of two polynomial expression. It is useful in Algebraic Mathematics. It takes two polynomials of degree 'm' and 'n' as input and prints their sum as output.*

Solution:

```
#include<stdio.h>

int main()
{
    int a[100],n,m,b[100],i,j;
    for(i=0;i<100;i++)
    {
        a[i]=0;
```

```
             b[i]=0;
        }
        printf("Enter the degree of polynomial 1\n");
        scanf("%d",&n);
        printf("Enter the polynomial 1\n");
        for(i=n;i>=0;i--)
        {
             printf("Enter the coefficient of x^%d : ",i);
             scanf("%d",&a[i]);       //input polynomial 1 of degree 'n'
        }
        printf("Enter the degree of polynomial 2\n");
        scanf("%d",&m);
        printf("Enter the polynomial 2\n");
        for(i=m;i>=0;i--)
        {
             printf("Enter the coefficient of x^%d : ",i);
             scanf("%d",&b[i]);       //input polynomial 1 of degree 'm'
        }
        for(i=(m>n)?m:n;i>=0;i--)    //print the sum
             printf("%dx^%d+ ",a[i]+b[i],i);
        printf("\n");


}
```

Output:

```
Enter the degree of polynomial 1
4
Enter the polynomial 1
Enter the coefficient of x^4 : 1
Enter the coefficient of x^3 : 2
Enter the coefficient of x^2 : 3
Enter the coefficient of x^1 : 4
Enter the coefficient of x^0 : 5
Enter the degree of polynomial 2
3
Enter the polynomial 2
Enter the coefficient of x^3 : 1
Enter the coefficient of x^2 : 2
Enter the coefficient of x^1 : 3
Enter the coefficient of x^0 : 4
1x^4+ 3x^3+ 5x^2+ 7x^1+ 9x^0
```

____

Problem 5: *Write a program to reverse all the elements of an array.*
Description: *The Programs aims at reversing all the elements of an array. It is useful in reversing a string and finding palindromic strings. It takes an array of size 'n' as input and prints the array in reverse order.*
Solution:
```
#include<stdio.h>

int main()
{
      int a[1000],n,i,j;
      printf("Enter the size of the array\n");
      scanf("%d",&n);      //input size of array
      printf("Enter the array\n");
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);       //input array
      printf("You have entered\n");
      for(i=0;i<n;i++)
            printf("%d ",a[i]);
      printf("\n");
      for(i=0;i<(n+1)/2;i++)   //reversing the array
      {
            j=a[i];
            a[i]=a[n-i-1];
            a[n-i-1]=j;
      }
      printf("Reversed Array:\n");
      for(i=0;i<n;i++)     //print reversed array
            printf("%d ",a[i]);
      printf("\n");
}
```
Output:
```
Enter the size of the array
5
Enter the array
1 2 3 4 5
You have entered
1 2 3 4 5
Reversed Array:
5 4 3 2 1
```
____

Problem 6: *Write a program to find out the transpose of a matrix.*
Description: *The Programs aims at finding the transpose of a matrix. It is useful in finding out the inverse of any matrix. It takes a matrix of size n*m as an input and prints its transpose as output.*

Solution:

```
#include<stdio.h>

int main()
{
      int a[100][100],m,n,i,j,b[100][100];
      printf("Enter the size of the matrix(m*n)\n");
      scanf("%d %d",&m,&n);    //input size of matrix
      printf("Enter the matrix:\n");
      for(i=0;i<m;i++)
            for(j=0;j<n;j++)
            {
                  scanf("%d",&a[i][j]);    //input matrix
                  b[j][i]=a[i][j];
            }
      printf("You have entered\n");
      for(i=0;i<m;i++)
      {
            for(j=0;j<n;j++)
            {
                  printf("%d ",a[i][j]);   //move its values
            }
            printf("\n");
      }
      printf("Transpose of the matrix is:\n");
      for(i=0;i<n;i++)
      {
            for(j=0;j<m;j++)
            {
                  printf("%d ",b[i][j]);   //print the transpose
            }
            printf("\n");
      }
}
```

Output:

```
Enter the size of the matrix(m*n)
3 4
Enter the matrix:
1 2 3 4 5 6 7 8 9 10 11 12
You have entered
1 2 3 4
```

```
5 6 7 8
9 10 11 12
Transpose of the matrix is:
1 5 9
2 6 10
3 7 11
4 8 12
```

==

Week 2

Problem 1: *WAP to create an array having n elements. Now create another array which is having n-i elements similar to the first array. Now find out the duplicates in the array.*
Description: *The Programs aims at finding duplicate elements in an array and removing them. It takes an array of size 'n' as input and prints the modified array as output.*
Solution:

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

int main()
{
     int n,i,j,*a,b[1000];
     printf("Enter the size of array: \n");
     scanf("%d",&n);      //input size of array
     a=(int*)malloc(n*sizeof(int));   //allocate memory
     printf("Enter the %d elements: \n",n);
     for(i=0;i<n;i++)
          scanf("%d",&a[i]);   //input array
    for(i=0;i<n;i++)         //processing a[i]
    {
        for(j=i+1;j<n;j++)
            b[j]=a[j];       //find duplicate of a[i]
        for(j=i+1;j<n;j++)
```

```
            if(a[i]==b[j])
            {
                if(a[j]==-32743)
                    continue;
                printf("Duplicate of %d found at %d index\n",a[i],j);
                a[j]=-32743;
            }
    }
}
```

Output:

```
Enter the size of array:
9
Enter the 9 elements:
1 2 1 3 2 4 5 3 3
Duplicate of 1 found at 2 index
Duplicate of 2 found at 4 index
Duplicate of 3 found at 7 index
Duplicate of 3 found at 8 index
```

Problem 2: *An array is having the element of size n, another array is also of size n having the index of the element. Now rearrange the order of first array as per the index.*
Eg: A[] ={2,3,4}
1. index[ ] ={0,1,2}
A[] = [3, 2, 4];
index[] = [1, 0, 2];

Description: *The Programs aims at rearranging a given array according to a given list of index. It takes an array of size 'n' and an array of indices of size 'n' as an input and it outputs the rearranged array.*

Solution:

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

int main()
{
      int n,i,*a,*b,*c;
      printf("Enter the size of array: \n");
      scanf("%d",&n);     //size of array input
      a=(int*)malloc(n*sizeof(int));
      b=(int*)malloc(n*sizeof(int));
      c=(int*)malloc(n*sizeof(int));
      printf("Enter the %d elements: \n",n);
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);  //array input
```

```
    printf("Enter the index no.: \n");
      for(i=0;i<n;i++)
           scanf("%d",&b[i]);  //index input
    for(i=0;i<n;i++)
        c[i]=a[b[i]];       //rearranging array 'a' into 'c'
    for(i=0;i<n;i++)
        a[i]=c[i];
    printf("\nNew Arranged array: \n");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);     //modified array 'a'
    printf("\n");
}
```

Output:

```
Enter the size of array:
7
Enter the 7 elements:
1 2 3 4 5 6 7
Enter the index no.:
5 6 0 2 1 4 3
New Arranged array:
6 7 1 3 2 5 4
```

Problem 3: *Find out the intersection and union of two arrays.*
Eg: arr1[]={7,1,5,2,3,6}
arr2[] = {3, 8, 6, 20, 7}
intersection: {3,6}
Description: *The Programs aims at finding intersection and union of two arrays. It may prove helpful in finding the duplicate elements in combination of two arrays. It takes two arrays of size 'm' and 'n' as input and prints their union and intersection.*
Solution:

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

int main()
{
      int n,m,i,j,k,l,*a,*b,*in,*un,uncount=0,incount=0;
      printf("Enter the size of array 1: \n");
      scanf("%d",&n);
      a=(int*)malloc(n*sizeof(int));
      printf("Enter the %d elements: \n",n);
      for(i=0;i<n;i++)        //input array 1
           scanf("%d",&a[i]);
```

```
printf("Enter the size of array 2: \n");
scanf("%d",&m);
b=(int*)malloc(m*sizeof(int));
printf("Enter the %d elements: \n",m);
for(i=0;i<m;i++)        //input array 2
     scanf("%d",&b[i]);
un=(int*)malloc((n+m)*sizeof(int));
for(i=0;i<n+m;i++) //merging 2 arrays
{
     if(i<n)
          un[i]=a[i];
     else
          un[i]=b[i-n];
}
for(i=0;i<n+m-uncount;i++)  //removing duplicacy from union
{
     for(j=i+1;j<n+m-uncount;j++)
     {
          if(un[i]==un[j])
          {
               for(k=j;k<n+m-1;k++)
               {
                    un[k]=un[k+1];
               }
               uncount++;
               j--;
          }
     }
}
printf("Union:\n{ ");
for(i=0;i<n+m-uncount;i++)  //printing union
     printf("%d, ",un[i]);
printf("}\n");
k=0;
in=(int*)malloc((n+m)*sizeof(int));
for(i=0;i<n;i++)    //finding common elements in both
{
     for(j=0;j<m;j++)
     {
          if(a[i]==b[j])
          {
               in[k++]=a[i];
               break;
```

```
                }
            }
        }
        for(i=0;i<k-incount;i++)        //removing duplicacy
        {
            for(j=i+1;j<k-incount;j++)
            {
                if(in[i]==in[j])
                {
                    for(l=j;l<k-1;l++)
                    {
                        in[l]=in[l+1];
                    }
                    incount++;
                    j--;
                }
            }
        }
        printf("Intersection:\n{ ");     //printing intersection
        for(i=0;i<k-incount;i++)
            printf("%d, ",in[i]);
        printf("}\n");
}
```

Output:

```
Enter the size of array 1:
5
Enter the 5 elements:
1 2 4 7 8
Enter the size of array 2:
7
Enter the 7 elements:
1 3 5 6 7 8 9
Union:
{ 1, 2, 4, 7, 8, 3, 5, 6, 9, }
Intersection:
{ 1, 7, 8, }
```

Problem 4: *An array of random number is given. Now find out all the 0's and put them in the last of the array.*
1. Arr1[]={7,1,0,2,0,6,7,5,0,3,0}
Arr1[]={7,1,2,6,7,5,3,0,0,0,0}

Description: *The Programs aims at finding all the 0 elements and putting them at the end of the array. This may prove helpful when we need to deal with 0 at last and give priority to the non zero elements. It takes an array of size 'n' as input and print the modified array with all the zeros at the last as output.*

Solution:

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

int main()
{
	int n,i,j,k,*a,uncount=0;
	printf("Enter the size of array: \n");
	scanf("%d",&n);
	a=(int*)malloc(n*sizeof(int));
	printf("Enter the %d elements: \n",n);
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);  //input array
	for(j=0;j<n-uncount;j++)
	{
		if(a[j]==0)
		{
			for(k=j;k<n-1;k++)
			{
				a[k]=a[k+1];    //delete zero from in between
			}
			a[n-uncount-1]=0;   //place zero at last
			uncount++;
			j--;
		}
	}
	printf("Modified Array: \n");
	for(i=0;i<n;i++)    //print modified array
		printf("%d ",a[i]);
	printf("\n");
}
```

Output:

```
Enter the size of array:
7
Enter the 7 elements:
1 0 2 0 0 3 4
Modified Array:
1 2 3 4 0 0 0
```

Problem 5: *Write a program to find all triplets in a sorted array that forms Geometric Progression(GP).*
Description: *The Programs aims at finding all the triplets which are in GP from a sorted array. It takes an array of size 'n' as an input and prints the count and the triplets which are there in the array.*
Solution:

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

int main()
{
      int a[1000],n,i,j,k,count=0;
      printf("Enter the size of array: \n");
      scanf("%d",&n);     //input array
      printf("Enter the %d elements of array: \n",n);
      for(i=0;i<n;i++)    //show existing array
            scanf("%d",&a[i]);
      for(i=0;i<n;i++)
      {
            for(j=i+1;j<n;j++)
            {
                  for(k=j+1;k<n;k++)
                  {
                        if(a[i]*a[k]==a[j]*a[j]) //condition for geometric mean
                        {
                              count++;
                              printf("%d , %d, %d\n",a[i],a[j],a[k]); //printing
those triplets
                        }
                  }
            }
      }
      printf("No. of triplets= %d\n",count);  //print total count of triplets
}
```

Output:

```
Enter the size of array:
9
Enter the 9 elements of array:
1 2 3 4 5 6 7 8 9
1 , 2, 4
1 , 3, 9
2 , 4, 8
4 , 6, 9
```

```
No. of triplets= 4
```

____

Problem 6: *Find the 2nd minimum distance between the 2 elements of array.*
Description: *The Programs aims at finding the 2nd minimum distance between 2 elements of array. It takes an array as an input and print the 2nd minimum distance of every element as output.*
Solution:

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

int main()
{
     int a[1000],b[1000],n,i,j,k,p;
     printf("Enter the size of array: \n");
     scanf("%d",&n);
     printf("Enter the %d elements of array: \n",n);
     for(i=0;i<n;i++)
          scanf("%d",&a[i]);  //array input
     for(i=0;i<n-1;i++)
     {
          for(j=i+1;j<n;j++)
          {
               b[j-(i+1)]=fabs(a[i]-a[j]);
          }
          for(p=0;p<n-(i+1);p++)
       {
           for(j=0;j<n-(i+1)-p-1;j++)
               if(b[j]>b[j+1])
               {
                    b[j]=b[j]+b[j+1];
                    b[j+1]=b[j]-b[j+1];
                    b[j]=b[j]-b[j+1];
               }
       }
       for(p=0;p<n-(i+1);p++)
       {
           if(b[p]==b[p+1])
               continue;
           else
               break;
       }
       k=b[p+1];
       if(i==n-2)
```

```
            k=b[p];
           printf("%d--> %d\n",a[i],k); //2nd min distance of each element
      }
}
```

Output:

```
Enter the size of array:
10
Enter the 10 elements of array:
1 4 9 16 25 36 49 64 81 100
1--> 8
4--> 12
9--> 16
16--> 20
25--> 24
36--> 28
49--> 32
64--> 36
81--> 19
```

Problem 7: *Write a program to find out the minimum sum up to 2 digit numbers from the elements of array, and also tell the count of numbers.*
Description: *The Programs aims at finding the minimum sum upto 2 digit from the elements of an array, we take an array as an input and print the desired count value and the elements' sum as output.*
Solution:

```
#include<stdio.h>

int main()
{
      int a[1000],n,i,sum=0,count=0;
      printf("Enter the size of array: \n");
      scanf("%d",&n); //input size of array
      printf("Enter the %d elements of array: \n",n);
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);  //input the array
      for(i=0;i<n;i++)
      {
            if(sum+a[i]<100)    //check if sum is less than 100
            {
                  sum+=a[i];
                  count++;
            }
        }                           //print sum and count
        printf("sum= %d, using no. of elements= %d\n",sum,count);
}
```

Output:

```
Enter the size of array:
10
Enter the 10 elements of array:
10 20 30 25 9 8 7 3 2 1
sum= 99, using no. of elements= 7
```

Problem 8: *Write a program to replace every element with the least greater element on its right. Any number will not be repeated.*

Description: *The Programs aims at replacing every element with the least greater element on its right indices. We take an array as an input and output the modified array with each element replaced with its least greater element on its right.*

Solution:

```
#include<stdio.h>
#include<math.h>
int small2(int b[],int j,int n)
{                   //function to find least greater on right indexes
        int i,temp,k=j;
        for(i=j;i<n;i++)
            for( ;j<n;j++)
            {
                if(b[j]>b[j+1])
                {
                    temp=b[j];
                    b[j]=b[j+1];
                    b[j+1]=temp;
                }
            }
        return b[k];
}


int main()
{
        int a[1000],b[1000],n,i,j,k,count=0,c[1000],p,temp,flag;
        printf("Enter the size of array: \n");
```

```
scanf("%d",&n);
printf("Enter the %d elements of array: \n",n);
for(i=0;i<n;i++)    //input array
{
     scanf("%d",&a[i]);
     c[i]=0;
}
for(i=0;i<n-1;i++)
{
     for(j=i+1;j<n;j++)
          b[j-(i+1)]=a[j];
  if(i==n-2)
      goto lab;
     for(p=0;p<n-i-1;p++)
      for(j=0 ;j<n-p-1;j++)
      {
          if(b[j]>b[j+1])
          {
              temp=b[j];
              b[j]=b[j+1];
              b[j+1]=temp;
          }
      }
  flag=0;
  lab:
  for(j=0;j<n-i-1;j++)
      if(a[i]<b[j])
      {
          flag=1;
          printf("%d ",b[j]);
          c[i]=1;
          break;
      }
  if(flag==0)
  {
      printf("%d ",a[i]);
      c[i]=1;
  }
}
for(i=0;i<n;i++)
{
     if(c[i]==0)
          printf("%d ",a[i]);
```

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

Output:

```
Enter the size of array:
10
Enter the 10 elements of array:
1 3 0 2 -1 8 9 13 12 19
2 8 2 8 8 9 12 19 19 19
```

Problem 9: *Write a program to print all subarrays with zero sum.*
Description: *The Programs aims at printing all the subarrays of an array whose sum is 0. It is helpful in finding the powerset of a given set of 'n' elements whose sum is 0. It takes an array as an input and prints all the subarrays of it in a new line.*
Solution:

```
#include<stdio.h>
#include<string.h>
char sa[5000][100];
int sai=0;
void print()    //print final value of subarrays after removing duplicacy
{
    int i,j,k,p,q,flag;
    for(i=0;i<sai;i++)
    {
        for(j=i+1;j<sai;j++)
            if(strcmp(sa[i],sa[j])==0)
        {
            sa[j][0]='\0';
        }
    }
    for(i=0;i<sai;i++)
    {
        flag=0;
            for(j=0;sa[i][j]!='\0';j++)
        {
            if(sa[i][j]=='\0')
                break;
            printf("%d ",(int)sa[i][j]-54);
            flag=1;
        }
        if(flag==1)
            printf("\n");
    }

}
void store(int a[],int n)   //store the subarrays whose sum is 0
{
   int i;
   for(i=0;i<n;i++)
        sa[sai][i]=(char)(a[i]+54);
    sa[sai][i]='\0';
   sai++;

}
int repeat(int a[],int n)   //recursive function to find subarrays
{
    /*if((n==1)&&(a[0]==0))
    {
        store(a,n);
        return 0;
    }*/
    if(n==1)
        return 0;
    int i,k,j=0,b[100],sum=0;
    for(j=0;j<n;j++)
    {
        k=0;
        for(i=0;i<n;i++)
        {
            if(i!=j)
                b[k++]=a[i];
        }
        repeat(b,n-1);  //recursive call
    }
    for(i=0;i<n-1;i++)
        sum+=b[i];
    if(sum==0)
    {
        store(b,n-1);
        return 0;
    }
}
```

```
int main()
{
      int a[1000],n,i,sum=0,j;
      printf("Enter the size of array: \n");
      scanf("%d",&n);
      printf("Enter the %d elements of array: \n",n);
      for(i=0;i<n;i++)
            scanf("%d",&a[i]);  //array input
     for(i=0;i<5000;i++)
         for(j=0;j<100;j++)
             sa[i][j]=3247;
      repeat(a,n);
      if(sai!=0)
         print();    //print subarrays
      else
         printf("Not possible\n");   //if no subarray is possible

}
```

Output:

```
Enter the size of array:
7
Enter the 7 elements of array:
-1 1 0 2 -2 3 -4
2 -2
0
0 2 -2
-1 -2 3
-1 0 -2 3
-1 1
-1 1 2 -2
-1 1 0
-1 1 0 2 -2
```

Problem 10: *Take the array which could be divided into two subarrays of equal sum.*
Description: *The Programs aims at dividing an array into two subarrays whose sum is equal. It may be helpful in dividing a fair-team-selection out of 'n' students whose marks are given, so that each team gets equal amount of talented students. It takes an array as input and prints the two subarrays of equal sum as output if it exists.*
Solution:

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

```
void print(int b[],int c[],int j,int k,int elb,int elc)
{       //function to print
    int i;
    printf("%d ",elc);
    for(i=0;i<j;i++)
    {
        if(b[i]==elb)
            continue;
        printf("%d ",b[i]);

    }
    printf("\nAND\n");
    printf("%d ",elb);
    for(i=0;i<k;i++)
    {
        if(c[i]==elc)
            continue;
        printf("%d ",c[i]);
    }
}

int main()
{
    int
a[1000],b[500],c[500],p,i,j=0,k=0,n,sum1=0,sum2=0,diff,count=0,sh,temp;
    time_t t;
    printf("Enter the size of array: \n");
    scanf("%d",&n);
    printf("Enter the %d elements: \n",n);
    for(i=0;i<n;i++)    //array input
    {
        scanf("%d",&a[i]);
        if(sum1>=sum2)
        {
            c[k++]=a[i];
            sum2+=a[i];
        }
        else
        {
            b[j++]=a[i];
            sum1+=a[i];
        }
```

```
}
diff=sum2-sum1; //diff of 2 subarrays
if(diff==0) //if diff becomes 0
{
    for(i=0;i<j;i++)
        printf("%d ",b[i]);
    printf("\nAND\n");
    for(i=0;i<k;i++)
        printf("%d ",c[i]);
}
else if(diff%2==0)  //if diff is even
{
    lab:
    for(i=0;i<j;i++)
    {
        for(p=0;p<k;p++)
        {
            if(c[p]-b[i]==diff/2)
            {
                print(b,c,j,k,b[i],c[p]);
                return 0;
            }

        }
    }
    count++;
    if(count>30000)
        return 0;
    if(count<15000)     //rearranging the array randomly
    {
        srand((unsigned) time(&t));
        sh=rand()%j;    //random function
        temp=b[sh];
        b[sh]=b[j-sh];
        b[j-sh]=temp;
    }
    else if(count>=15000)
    {
        srand((unsigned) time(&t));
        sh=rand()%k;        //random function
        temp=c[sh];
        c[sh]=c[k-sh];
        c[k-sh]=temp;
        }
        goto lab;
    }
}
```

Output:

```
Enter the size of array:
10
Enter the 10 elements:
1 2 3 4 0 2 3 1 2 8
8 2 1 2
AND
4 1 3 0 2 3
```

===

Week 3

Problem 1: *Insert a node in the linked list*
a. At the beginning
b. At the end
c. At a particular position.
Apply switch case for the options.
Description: *The Programs aims at creation of a linked list and inserting nodes in it at runtime. This may prove helpful in managing public records in schools, hospitals, etc. We take a linked list of size 'n' as input and give options for inserting a new node in linked list.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k;
      struct node *start,*ptr,*temp,*temp2;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)         //input initial linked list
      {
            if(i==0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for 1st node: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  start=ptr;
                  printf("1st node constructed\n");
            }
            else if(i>0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for %d node: \n",i+1);
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  if(i==1)
                        start->link=ptr;
                  else
                        temp->link=ptr;
                  temp=ptr;
            }

      }
      while(n)         //infinite loop for asking user the options
      {
      printf("Please Enter your choice: \n");
      printf(" press 1 for insertion at beg.: \n");
      printf(" Press 2 for insertion at particular position: \n");
      printf(" Press 3 for insertion at last.: \n");
      printf(" Press 0 to exit the program: \n");
      printf(" Press 4 to print existing list\n");
      scanf("%d",&n);
      if(n==0)
            break;
      switch(n)
      {
            case 1:                  //insert at begin
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data to be inserted: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=start;
                  start=ptr;
                  printf("Node created\n_____________________________\n");
                  break;
            case 2:                  //insert at kth position
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the position where you want to insert: \n");
                  scanf("%d",&k);
                  printf("Enter the data to be inserted: \n");
                  scanf("%d",&ptr->data);
                  temp=start;
                  for(i=0;i<k-2;i++)
```

```
                {
                     temp=temp->link;
                }
                temp2=temp->link;
                temp->link=ptr;
                ptr->link=temp2;
                printf("Node created\n_____________________________\n");
                break;
           case 3:                      //insert at last
                ptr=(struct node*)malloc(sizeof(struct node));
                printf("Enter the data to be inserted: \n");
                scanf("%d",&ptr->data);
                temp=start;
                while(temp->link!=NULL)
                {
                     temp=temp->link;
                }
                temp->link=ptr;
                ptr->link=NULL;
                printf("Node created\n_____________________________\n");
                break;
           case 4:                  //print the current list
                temp=start;
                while(temp!=NULL)
                {
                     printf("%d ",temp->data);
                     temp=temp->link;
                }
                printf("\n");
                printf("_____________________________\n");
                break;
      }
      }

}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
```

```
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
1
Enter the data to be inserted:
0
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
4
0 1 2 3 4 5 6 7
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
2
Enter the position where you want to insert:
3
Enter the data to be inserted:
```

```
-1
Node created
______________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
4
0 1 -1 2 3 4 5 6 7
______________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
3
Enter the data to be inserted:
8
Node created
______________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
4
0 1 -1 2 3 4 5 6 7 8
______________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
0
```

Problem 2: *Implement a function to reverse a linked list, and return the head of the reversed list.*

Description: *The Programs aims at creating a linked list of n nodes and reversing the linkage if all the nodes thereby reversing the linked list. It proves helpful in finding out palindromic list. We take a linked list as as input and print the reverse of it as output.*

Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k;
      struct node *start,*start1,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)            //linked list input
      {
            if(i==0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for 1st node: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  start=ptr;
                  printf("1st node constructed\n");
            }
            else if(i>0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for %d node: \n",i+1);
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  if(i==1)
                        start->link=ptr;
                  else
                        temp->link=ptr;
                  temp=ptr;
            }
```

```
    }
    printf("Reversed linked list: \n");
    ptr=start;
    while(ptr->link!=NULL)
         ptr=ptr->link;
    start1=ptr;
    temp=ptr;
    lab2:
    ptr=start;
    while(ptr->link!=temp)  //reversing linked list
    {
         ptr=ptr->link;

    }
    temp->link=ptr;
         temp=ptr;
    if(ptr!=start)
         goto lab2;
    start->link=NULL;
    ptr=start1;
    while(ptr!=NULL)        //print reversed linked list
    {
         printf("%d ",ptr->data);
         ptr=ptr->link;
    }


}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
```

```
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
Reversed linked list:
7 6 5 4 3 2 1
```

____

Problem 3: *Delete an element from particular position.*
Description: *The Programs aims at creating a linked list and deleting k'th node from that list. It is useful in deleting those record which are expired or are no longer valid. It takes a linked list as input and provide option for deleting k'th element and prints the updated list as output.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k;
      struct node *start,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)            //linked list input
      {
            if(i==0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for 1st node: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  start=ptr;
                  printf("1st node constructed\n");
            }
            else if(i>0)
```

```
        {
              ptr=(struct node*)malloc(sizeof(struct node));
              printf("Enter the data for %d node: \n",i+1);
              scanf("%d",&ptr->data);
              ptr->link=NULL;
              if(i==1)
                    start->link=ptr;
              else
                    temp->link=ptr;
              temp=ptr;
        }

  }
  while(n)
  {
  printf("Please Enter your choice: \n");
  printf(" Press 1 for deletion at particular position: \n");
  printf(" Press 0 to exit the program: \n");
  printf(" Press 2 to print existing list\n");
  scanf("%d",&n);
  if(n==0)
        break;
  switch(n)
  {

        case 1:          //performing deletion
              printf("Enter the position where you want to delete: \n");
              scanf("%d",&k);
              if(k==1)            //primitive case
              {
                    start=start->link;
                    goto lab;
              }
              temp=start;     //rest of the cases
              for(i=0;i<k-2;i++)
              {
                    temp=temp->link;
              }
              temp2=temp->link;
              temp3=temp2->link;
              temp->link=temp3;
              lab:
              printf("Node deleted\n______________________________\n");
```

```
                break;
           case 2:
                temp=start;     //printing the list
                while(temp!=NULL)
                {
                     printf("%d ",temp->data);
                     temp=temp->link;
                }
                printf("\n");
                printf("_____________________________\n");
                break;
      }
      }

}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
Please Enter your choice:
 Press 1 for deletion at particular position:
 Press 0 to exit the program:
 Press 2 to print existing list
1
Enter the position where you want to delete:
1
Node deleted
```

```
_____________________________
Please Enter your choice:
 Press 1 for deletion at particular position:
 Press 0 to exit the program:
 Press 2 to print existing list
2
2 3 4 5 6 7

_____________________________
Please Enter your choice:
 Press 1 for deletion at particular position:
 Press 0 to exit the program:
 Press 2 to print existing list
1
Enter the position where you want to delete:
6
Node deleted

_____________________________
Please Enter your choice:
 Press 1 for deletion at particular position:
 Press 0 to exit the program:
 Press 2 to print existing list
2
2 3 4 5 6

_____________________________
Please Enter your choice:
 Press 1 for deletion at particular position:
 Press 0 to exit the program:
 Press 2 to print existing list
0
```

==

Week 4

Problem 1: *Insert and delete a node in the doubly linked list.*
Description: *The Programs aims at creating a doubly linked list with options to modify the nodes by insertion and deletion at further stage. The advantage of using doubly linked list is that we can traverse in both directions, thus reducing time complexity.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
      struct node *prev;
};

int main()
{
      int n=1,i,m,k;
      struct node *start,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
```

```
scanf("%d",&m);
for(i=0;i<m;i++)        //input the linked list
{
      if(i==0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for 1st node: \n");
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            ptr->prev=NULL;
            start=ptr;
            temp=start;
            printf("1st node constructed\n");
      }
      else if(i>0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            ptr->prev=temp;
            temp->link=ptr;
            temp=ptr;
      }

}
while(n)        //infinite loop for giving options
{
printf("Please Enter your choice: \n");
printf(" press 1 for insertion at beg.: \n");
printf(" Press 2 for insertion at particular position: \n");
printf(" Press 3 for insertion at last.: \n");
printf(" Press 0 to exit the program: \n");
printf(" Press 4 to print existing list\n");
printf(" Press 5 for deletion at particular position: \n");
scanf("%d",&n);
if(n==0)
      break;
switch(n)
{
      case 1:         //insertion at beg
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data to be inserted: \n");
```

```
        scanf("%d",&ptr->data);
        ptr->prev=NULL;
        ptr->link=start;
        start->prev=ptr;
        start=ptr;
        printf("Node created\n______________________________\n");
        break;
    case 2:     //insertion at k
        ptr=(struct node*)malloc(sizeof(struct node));
        printf("Enter the position where you want to insert: \n");
        scanf("%d",&k);
        printf("Enter the data to be inserted: \n");
        scanf("%d",&ptr->data);
        temp=start;
        for(i=0;i<k-2;i++)
        {
            temp=temp->link;
        }
        temp2=temp->link;
        temp->link=ptr;
        ptr->prev=temp;
        ptr->link=temp2;
        temp2->prev=ptr;
        printf("Node created\n______________________________\n");
        break;
    case 3:     //insertion at last
        ptr=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data to be inserted: \n");
        scanf("%d",&ptr->data);
        temp=start;
        while(temp->link!=NULL)
        {
            temp=temp->link;
        }
        temp->link=ptr;
        ptr->prev=temp;
        ptr->link=NULL;
        printf("Node created\n______________________________\n");
        break;
    case 4:     //print the list
        temp=start;
        while(temp!=NULL)
        {
```

```
                    printf("%d ",temp->data);
                    temp=temp->link;
               }
               printf("\n");
               printf("_____________________________\n");
               break;
          case 5:         //deletion at kth position
               printf("Enter the position where you want to delete: \n");
               scanf("%d",&k);
               if(k==1)
               {
                    start=start->link;
                    start->prev=NULL;
                    goto lab;
               }
               temp=start;
               for(i=0;i<k-2;i++)
               {
                    temp=temp->link;
               }
               temp2=temp->link;
               if(temp2->link!=NULL)
               {
                    temp3=temp2->link;
                    temp->link=temp3;
                    temp3->prev=temp;
               }
               else
                    temp->link=NULL;
               temp2->prev=temp2->link=NULL;
               lab:
               printf("Node deleted\n_____________________________\n");
               break;
          default :
               printf("Invalid Input\n");
     }
     }

}
```

Output:

```
Enter the no. of elements to be created at first
5
```

```
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
1
Enter the data to be inserted:
0
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 1 2 3 4 5
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
2
Enter the position where you want to insert:
3
```

```
Enter the data to be inserted:
-1
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 1 -1 2 3 4 5
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
3
Enter the data to be inserted:
6
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 1 -1 2 3 4 5 6
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
```

```
5
Enter the position where you want to delete:
4
Node deleted
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 1 -1 3 4 5 6
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
0
```

Problem 2: *Insert and delete a node in the circular linked list.*
Description: *The Programs aims at creating a circular linked list and giving options for modifying the nodes at a later stage. The advantage of circular linked list is that the last node is directly connected to the head node.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k;
      struct node *start,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
```

```
scanf("%d",&m);
for(i=0;i<m;i++)        //linked list input
{
      if(i==0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for 1st node: \n");
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            start=ptr;
            temp=start;
            printf("1st node constructed\n");
      }
      else if(i>0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            temp->link=ptr;
            temp=ptr;
      }

}
ptr->link=start;        //link last node with head
while(n)
{
printf("Please Enter your choice: \n");
printf(" press 1 for insertion at beg.: \n");
printf(" Press 2 for insertion at particular position: \n");
printf(" Press 3 for insertion at last.: \n");
printf(" Press 0 to exit the program: \n");
printf(" Press 4 to print existing list\n");
printf(" Press 5 for deletion at particular position: \n");
scanf("%d",&n);
if(n==0)
      break;
switch(n)
{
      case 1:     //insertion at beg
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data to be inserted: \n");
            scanf("%d",&ptr->data);
```

```
        temp=start;
        while(temp->link!=start)
             temp=temp->link;
        ptr->link=start;
        start=ptr;
        temp->link=start;
        printf("Node created\n______________________________\n");
        break;
   case 2:     //insertion at k
        ptr=(struct node*)malloc(sizeof(struct node));
        printf("Enter the position where you want to insert: \n");
        scanf("%d",&k);
        printf("Enter the data to be inserted: \n");
        scanf("%d",&ptr->data);
        temp=start;
        for(i=0;i<k-2;i++)
        {
             temp=temp->link;
        }
        temp2=temp->link;
        temp->link=ptr;
        ptr->link=temp2;
        printf("Node created\n______________________________\n");
        break;
   case 3:         //insertion at last
        ptr=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data to be inserted: \n");
        scanf("%d",&ptr->data);
        temp=start;
        while(temp->link!=start)
        {
             temp=temp->link;
        }
        temp->link=ptr;
        ptr->link=start;
        printf("Node created\n______________________________\n");
        break;
   case 4:     //print the list
        temp=start;
        while(1)
        {
             printf("%d ",temp->data);
             temp=temp->link;
```

```
                    if(temp==start)
                         break;
               }
               printf("\n");
               printf("______________________________\n");
               break;
          case 5:        //deletion at kth
               printf("Enter the position where you want to delete: \n");
               scanf("%d",&k);
               temp=start;
               if(k==1)
               {
                    while(temp->link!=start)
                         temp=temp->link;
                    start=start->link;
                    temp->link=start;
                    goto lab;
               }
               temp=start;
               for(i=0;i<k-2;i++)
               {
                    temp=temp->link;
               }
               temp2=temp->link;
               if(temp2->link!=start)
               {
                    temp3=temp2->link;
                    temp->link=temp3;
               }
               else
                    temp->link=start;
               temp2->link=NULL;
               lab:
               printf("Node deleted\n______________________________\n");
               break;
          default :
               printf("Invalid Input\n");
     }
     }

}
```

Output:

```
Enter the no. of elements to be created at first
5
Enter the data for 1st node:
10
1st node constructed
Enter the data for 2 node:
20
Enter the data for 3 node:
30
Enter the data for 4 node:
40
Enter the data for 5 node:
50
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
1
Enter the data to be inserted:
0
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 10 20 30 40 50
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
2
```

```
Enter the position where you want to insert:
3
Enter the data to be inserted:
15
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 10 15 20 30 40 50
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
3
Enter the data to be inserted:
60
Node created
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 10 15 20 30 40 50 60
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
```

```
 Press 4 to print existing list
 Press 5 for deletion at particular position:
5
Enter the position where you want to delete:
5
Node deleted
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
4
0 10 15 20 40 50 60
_____________________________
Please Enter your choice:
 press 1 for insertion at beg.:
 Press 2 for insertion at particular position:
 Press 3 for insertion at last.:
 Press 0 to exit the program:
 Press 4 to print existing list
 Press 5 for deletion at particular position:
0
```

Problem 3: *Find the middle node of the list and delete it.*
Description: *The Programs aims at creating linked list of n nodes and deleting its middle node. We will use the most fastest algorithm to find out the middle most element.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k,count=1;
```

```
 struct node *start,*ptr,*temp,*temp2,*temp3;
 start=NULL;
 printf("Enter the no. of elements to be created at first\n");
 scanf("%d",&m);
 for(i=0;i<m;i++)
 {                    //linked list input
      if(i==0)
      {
           ptr=(struct node*)malloc(sizeof(struct node));
           printf("Enter the data for 1st node: \n");
           scanf("%d",&ptr->data);
           ptr->link=NULL;
           start=ptr;
           temp=start;
           printf("1st node constructed\n");
      }
      else if(i>0)
      {
           ptr=(struct node*)malloc(sizeof(struct node));
           printf("Enter the data for %d node: \n",i+1);
           scanf("%d",&ptr->data);
           ptr->link=NULL;
           temp->link=ptr;
           temp=ptr;
      }

 }
 ptr=start;
 temp=start;
 temp2=start;
 while(1)
{       //algo for finding middlemost element
    if(ptr->link==NULL)
        break;
    if(ptr->link->link==NULL)
        break;
    temp2=temp;
    temp=temp->link;
    ptr=ptr->link->link;
}
temp2->link=temp->link;     //deleting the middle element
temp->link=NULL;
 printf("After deleting the Middle elements,The linked list becomes: \n");
```

```
      ptr=start;
      while(ptr!=NULL)
      {            //print modified list
            printf("%d ",ptr->data);
            ptr=ptr->link;
      }
      printf("\n");


}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
After deleting the Middle elements,The linked list becomes:
1 2 3 5 6 7
```

==

Week 5

Problem 1: *Find the middle element of the circular linked list.*

Description: *The Programs aims at creating circular linked list of n nodes and finding its middle node. We will use the most fastest algorithm to find out the middle most element.*

Solution:

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

struct node     //self referencial structure
```

```
{
     int data;
     struct node *link;
};

int main()
{
     int i,m,k;
     struct node *start,*ptr,*temp;
     start=NULL;
     printf("Enter the no. of elements to be created at first\n");
     scanf("%d",&m);
     for(i=0;i<m;i++)        //input linked list
     {
          if(i==0)
          {
               ptr=(struct node*)malloc(sizeof(struct node));
               printf("Enter the data for 1st node: \n");
               scanf("%d",&ptr->data);
               ptr->link=NULL;
               start=ptr;
               temp=start;
               printf("1st node constructed\n");
          }
          else if(i>0)
          {
               ptr=(struct node*)malloc(sizeof(struct node));
               printf("Enter the data for %d node: \n",i+1);
               scanf("%d",&ptr->data);
               ptr->link=NULL;
               temp->link=ptr;
               temp=ptr;
          }

     }
     ptr->link=start;        //linking last node with head
     ptr=start;
     temp=start;
     while(1)
    {       //algo for mid element
       if(ptr->link==start)
           break;
       if(ptr->link->link==start)
```

```
            break;
        temp=temp->link;
        ptr=ptr->link->link;
    }
     printf("Middle element of the circular linked list is :
%d\n",temp->data);

}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
Middle element of the circular linked list is : 4
```

Problem 2: *Find the 2nd last element of the singly linked list.*
Description: *The Programs aims at creating a single linked list and finding the 2nd last element of that list in O(n) complexity.*
Solution:

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

struct node     //self referencial structure
{
     int data;
```

```
    struct node *link;
};

int main()
{
    int i,m,k,count=1;
    struct node *start,*ptr,*temp;
    start=NULL;
    printf("Enter the no. of elements to be created at first\n");
    scanf("%d",&m);
    for(i=0;i<m;i++)        //input linked list
    {
        if(i==0)
        {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for 1st node: \n");
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            start=ptr;
            temp=start;
            printf("1st node constructed\n");
        }
        else if(i>0)
        {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            temp->link=ptr;
            temp=ptr;
        }

    }
    if(m==1)        //primitive case
        printf("2nd last element not found\n");
    else
    {
    ptr=start;
    temp=start->link;
    while(temp->link!=NULL)     //traverse till last
    {
        temp=temp->link;
        ptr=ptr->link;
```

```
      }           //print 2nd last
      printf("2nd last element of the linked list is : %d\n",ptr->data);
      }

}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the data for 7 node:
7
2nd last element of the linked list is : 6
```

____

Problem 3: *Add two polynomials using suitable linked list.*
Description: *The Programs aims at adding two mathematical polynomial expression as an application of linked list. The coefficients of two terms of same exponents can be added.*
Solution:

```
#include<stdio.h>
#include<malloc.h>
#include<conio.h>
struct link{        //self referencial structure
       int coeff;
       int pow;
       struct link *next;
       };
struct link *poly1=NULL,*poly2=NULL,*poly=NULL;
void create(struct link *node)
{           //create a polynomial using list
 char ch;
 do
 {
  printf("\n enter coeff:");
  scanf("%d",&node->coeff);
  printf("\n enter power:");
  scanf("%d",&node->pow);
  node->next=(struct link*)malloc(sizeof(struct link));
  node=node->next;
  node->next=NULL;
  printf("\n continue(y/n):");
  ch=getch();
 }
 while(ch=='y' || ch=='Y');
}
void show(struct link *node)
{               //to print a linked list
 while(node->next!=NULL)
 {
  printf("%dx^%d",node->coeff,node->pow);
  node=node->next;
  if(node->next!=NULL)
   printf("+");
 }
}
void polyadd(struct link *poly1,struct link *poly2,struct link *poly)
{           //adding two polynomial
     while(poly1->next &&  poly2->next)
     {
      if(poly1->pow>poly2->pow)
      {
       poly->pow=poly1->pow;
       poly->coeff=poly1->coeff;
       poly1=poly1->next;
       }
      else if(poly1->pow<poly2->pow)
      {
       poly->pow=poly2->pow;
       poly->coeff=poly2->coeff;
       poly2=poly2->next;
       }
      else
      {
       poly->pow=poly1->pow;
       poly->coeff=poly1->coeff+poly2->coeff;
```

```
         poly1=poly1->next;
         poly2=poly2->next;
          }
        poly->next=(struct link *)malloc(sizeof(struct link));
        poly=poly->next;
        poly->next=NULL;
       }
       while(poly1->next || poly2->next)
       {
        if(poly1->next)
        {
         poly->pow=poly1->pow;
         poly->coeff=poly1->coeff;
         poly1=poly1->next;
          }
        if(poly2->next)
        {
         poly->pow=poly2->pow;
         poly->coeff=poly2->coeff;
         poly2=poly2->next;
          }
         poly->next=(struct link *)malloc(sizeof(struct link));
         poly=poly->next;
         poly->next=NULL;
          }
}
main()
{
        char ch;
        do{
        poly1=(struct link *)malloc(sizeof(struct link));
        poly2=(struct link *)malloc(sizeof(struct link));
        poly=(struct link *)malloc(sizeof(struct link));
        printf("\nenter 1st number:");
        create(poly1);
        printf("\nenter 2nd number:");
        create(poly2);
        printf("\n1st Number:");
        show(poly1);
        printf("\n2nd Number:");
        show(poly2);
        polyadd(poly1,poly2,poly);
        printf("\nAdded polynomial:");
```

```
      show(poly);
      printf("\n add two more numbers:");
      ch=getch();
      }
      while(ch=='y' || ch=='Y');
}
```

Output:

```
enter 1st number:
 enter coeff:4

 enter power:3

 continue(y/n):y
 enter coeff:
4

 enter power:2

 continue(y/n):y
 enter coeff:4

 enter power:1

 continue(y/n):y
 enter coeff:4

 enter power:0

 continue(y/n):n
enter 2nd number:
 enter coeff:5

 enter power:2

 continue(y/n):y
 enter coeff:5

 enter power:1

 continue(y/n):y
 enter coeff:5
 enter power:0

 continue(y/n):n
1st Number:4x^3+4x^2+4x^1+4x^0
2nd Number:5x^2+5x^1+5x^0
Added polynomial:4x^3+9x^2+9x^1+9x^0
 add two more numbers:n
```

____

Problem 4: *Find out the union and intersection of two list.*

Description: *The Programs aims at finding union and intersection of two linked list. It may prove helpful in finding the duplicate elements in combination of two lists. It takes two lists of size 'm' and 'n' as input and prints their union and intersection.*

Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n,i,m,k,count=1;
      struct node *start1,*start2,*ptr,*temp,*temp2,*temp3;
      start1=NULL;         //head of list1
      start2=NULL;         //head of list2
      printf("Enter the no. of elements to be created in 1st list\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)          //input list1
      {
            if(i==0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for 1st node: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  start1=ptr;
                  temp=start1;
                  printf("1st node constructed\n");
```

```
        }
        else if(i>0)
        {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            temp->link=ptr;
            temp=ptr;
        }

    }
    printf("Enter the no. of elements to be created in 2nd list\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)        //input list2
    {
        if(i==0)
        {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for 1st node: \n");
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            start2=ptr;
            temp=start2;
            printf("1st node constructed\n");
        }
        else if(i>0)
        {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            temp->link=ptr;
            temp=ptr;
        }

    }
    ptr=start1;
    printf("Union of these 2 list:\n");
    while(ptr!=NULL)        //finding union
    {
        printf("%d  ",ptr->data);
        ptr=ptr->link;
        }
        ptr=start2;
        while(ptr!=NULL)
        {
                printf("%d  ",ptr->data);
                ptr=ptr->link;
        }
        printf("\n");
        ptr=start1;
        printf("Intersection of the 2 lists:\n");
        while(ptr!=NULL)        //finding intersection
        {
                temp2=start2;
                while(temp2!=NULL)
                {
                        if(temp2->data==ptr->data)
                                printf("%d  ",ptr->data);
                        temp2=temp2->link;
                }
                ptr=ptr->link;
        }
        printf("\n");
        return 0;
}
```

Output:

```
Enter the no. of elements to be created in 1st list
6
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
5
Enter the data for 6 node:
6
Enter the no. of elements to be created in 2nd list
```

```
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
0
Enter the data for 3 node:
2
Enter the data for 4 node:
7
Enter the data for 5 node:
8
Enter the data for 6 node:
9
Enter the data for 7 node:
11
Union of these 2 list:
1  2  3  4  5  6  1  0  2  7  8  9  11
Intersection of the 2 lists:
1  2
```

Problem 5: *Find out whether a singly linked list is palindrome or not.*
Description: *The Programs aims at finding out whether a linked list is palindrome or not. Palindromes are same even if they are reversed.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k,flag=1;
      struct node *start,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)        //input linked list
```

```
{
      if(i==0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for 1st node: \n");
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            start=ptr;
            temp=start;
            printf("1st node constructed\n");
      }
      else if(i>0)
      {
            ptr=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data for %d node: \n",i+1);
            scanf("%d",&ptr->data);
            ptr->link=NULL;
            temp->link=ptr;
            temp=ptr;
      }

}
temp=ptr=start;
if(m==1)         //primitive case
{
      printf("Palindromic list\n");
      return 0;
}
while(ptr->link!=NULL)
      ptr=ptr->link;
while(temp!=NULL)        //palindrome checking
{
      if(temp->data!=ptr->data)
      {
            flag=0;
            break;
      }
      if((temp->link==ptr)||(temp==ptr))
            break;
      temp2=start;
      while(temp2->link!=ptr)
            temp2=temp2->link;
      ptr=temp2;
```

```
            temp=temp->link;
      }
      if(flag==0)      //printing message
            printf("Not a palindromic list\n");
      else
            printf("Palidromic list\n");
      return 0;

}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
1st node constructed
Enter the data for 2 node:
2
Enter the data for 3 node:
3
Enter the data for 4 node:
4
Enter the data for 5 node:
3
Enter the data for 6 node:
2
Enter the data for 7 node:
1
Palindromic list
```

___

Problem 6: *Write a program to remove duplicate nodes from a linked list.*
Description: *The Programs aims at removing duplicates nodes from a linked list. It may prove helpful in cleaning records from reduntant and useless data such as duplicates of data items which are already present. It takes a list as input and prints the modified list as output.*
Solution:

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

struct node     //self referencial structure
{
      int data;
      struct node *link;
};

int main()
{
      int n=1,i,m,k,count=1;
      struct node *start,*ptr,*temp,*temp2,*temp3;
      start=NULL;
      printf("Enter the no. of elements to be created at first\n");
      scanf("%d",&m);
      for(i=0;i<m;i++)        //linked list input
      {
            if(i==0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for 1st node: \n");
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  start=ptr;
                  temp=start;
                  printf("1st node constructed\n");
            }
            else if(i>0)
            {
                  ptr=(struct node*)malloc(sizeof(struct node));
                  printf("Enter the data for %d node: \n",i+1);
                  scanf("%d",&ptr->data);
                  ptr->link=NULL;
                  temp->link=ptr;
                  temp=ptr;
            }

      }
      ptr=start;
      printf("Before removing duplicates:\n");
      while(ptr!=NULL)        //show linked list
      {
            printf("%d  ",ptr->data);
            ptr=ptr->link;
      }
      printf("\n");
      if(m==1)        //primitive case
```

```
    {
        printf("After removing duplicates: \n %d\n",start->data);
        return 0;
    }
    ptr=start;
    while(ptr!=NULL)        //algo for duplicacy removal
    {
        temp=ptr->link;
        while(temp!=NULL)
        {
            if(ptr->data==temp->data)
            {
                temp2=ptr;
                while(temp2->link!=temp)
                    temp2=temp2->link;
                temp2->link=temp->link;
                temp3=temp->link;
                temp->link=NULL;
                temp=temp3;

            }
            else
            temp=temp->link;
        }
        ptr=ptr->link;
    }
    ptr=start;
    printf("After removing duplicates:\n");
    while(ptr!=NULL)        //modified list
    {
        printf("%d  ",ptr->data);
        ptr=ptr->link;
    }
    printf("\n");
    return 0;
}
```

Output:

```
Enter the no. of elements to be created at first
7
Enter the data for 1st node:
1
```

```
1st node constructed
Enter the data for 2 node:
1
Enter the data for 3 node:
2
Enter the data for 4 node:
2
Enter the data for 5 node:
3
Enter the data for 6 node:
2
Enter the data for 7 node:
3
Before removing duplicates:
1  1  2  2  3  2  3
After removing duplicates:
1  2  3
```

==

Week 6

Problem Statement:
Create Function for each question. And call it using switch case.(eg: case1: Q1) .
Presentation matters.
1. Implement PUSH() in stack using array
2. Implement POP() from the stack using array.
3. Implement HeadRecursion() and TailRecursion().
4. Implement Infix2Postfix() function which will convert an infix expression to postfix.
5. Implement Print(), which will print all the above functions.

Description: *The Programs aims at implementing stack, recursion and infix to postfix evaluation.*
STACK: Expression evaluation and syntax parsing. Calculators employing reverse Polish notation use a ***stack*** *structure to hold values. Expressions can be represented in prefix, postfix or infix notations and conversion from one form to another may be accomplished using a* ***stack****.*
RECURSION: ***Recursion*** *is a programming technique in which a method makes a call to itself to solve a particular problem. Such methods are called* ***recursive****.* ***Recursion*** *is a programming technique whose correct usage leads to elegant solutions to certain problems like Tower Of Hanoi etc.*
NOTATIONS (POSTFIX): ***Postfix*** *notation, also known as RPN, is very easy to process left-to-right. An operand is pushed onto a stack; an operator pops its operand(s) from the stack and pushes the result. Little or no parsing is necessary.*

Solution:
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int top=-1,*st,n;
int push()      //function to push into stack
{
    top++;
    if(top==n)
    {
        printf("Overflow of stack.\n");
        top--;
        return 0;
    }
    printf("Enter the element you want to push\n");
    scanf("%d",&st[top]);
    return 0;
}
int pop()       //function to pop out of stack
{
    if(top==-1)
    {
        printf("Underflow of stack\n");
        return 0;
    }
    printf("%d is popped out of stack\n",st[top]);
    top--;
    return 0;
}
int print()     //function to print stack
{
    int i;
    if(top==-1)
    {
        printf("Underflow of stack. | Stack is empty\n");
        return 0;
    }
    printf("\n\nCurrent status of stack:\n");
    for(i=0;i<=top;i++)
        printf("%d  ",st[i]);
    printf("\n");
    return 0;
}
int sumhead(int n)      //head recursion
{
    int sum;
    if (n!=0)
    {
        printf("+%d",n);
        sum=n+sumhead(n-1); // sum() function calls itself
    }
    else
    {
        printf("=");
        return n;
    }
    return sum;
}
int sumtail(int n)      //tail recursion
```

```
{
    int sum;
    if (n!=0)
    {
        sum=n+sumtail(n-1); // sum() function calls itself
        if(n==1) printf("%d",n);
        else printf("+%d",n);
    }
    else
    {
        return n;
    }
    return sum;
}
int infix2postfix()     //function for infix to postfix
{
    char a[1000][10],stack[1000],c[1000],d;
    int b[1000],i,j=0,p=0,len,t=-1;
    printf("Enter an infix expression (USE $ TO TERMINATE):\n");
    for(i=0; ;i++)
    {
        scanf("%s",a[i]);
        if(a[i][0]=='$')
            break;
    }
    a[i][0]='\0';
    len=i;
    i=0;
    while(a[i][0]!='\0')         //various cases for infix to postfix
    {

if((a[i][0]=='*')||(a[i][0]=='/')||(a[i][0]=='%')||(a[i][0]=='+')||(a[i][0]=='
-')||(a[i][0]=='(')||(a[i][0]==')')||(a[i][0]=='^'))
        {
            if(t==-1)
            {
                t++;
                stack[t]=a[i][0];
            }
            else if(((stack[t]=='(')||(a[i][0]=='('))&&(a[i][0]!=')'))
            {
                t++;
                stack[t]=a[i][0];
            }
            else if((stack[t]=='^')&&(a[i][0]=='^'))
            {
                printf("%c ",stack[t]);
                t--;
                t++;
                stack[t]=a[i][0];
            }
            else if((a[i][0]=='^')&&(stack[t]!='^'))
            {
                t++;
                stack[t]=a[i][0];
            }
            else if((stack[t]=='^')&&(a[i][0]!=')'))
            {
                printf("%c ",stack[t]);
                t--;
                if(t>-1)
                if(((a[i][0]=='-')||(a[i][0]=='+'))&&(stack[t]!='('))
                {
                    printf("%c ",stack[t]);
                    t--;
                    if(t>-1)
                        if((stack[t]=='-')||(stack[t]=='+'))
                        {
                        printf("%c ",stack[t]);
                        t--;
                        }
                }
                if(t>-1)

if(((a[i][0]=='/')||(a[i][0]=='*')||(a[i][0]=='%'))&&((stack[t]=='/')||(stack[
t]=='*')||(stack[t]=='%')))
                {
                    printf("%c ",stack[t]);
                    t--;
                }
                t++;
                stack[t]=a[i][0];
            }
            else
if(((stack[t]=='-')||(stack[t]=='+'))&&((a[i][0]=='%')||(a[i][0]=='*')||(a[i][
0]=='/')||(a[i][0]=='^')))
```

```
            {
                t++;
                stack[t]=a[i][0];
            }
            else
if(((stack[t]=='%')||(stack[t]=='/')||(stack[t]=='*')||(a[i][0]=='^'))&&(a[i][
0]!=')')&&(a[i][0]!='^'))
            {
                printf("%c ",stack[t]);
                t--;
                j=t;
                if((a[i][0]=='+') || (a[i][0]=='-'))
                    if((stack[j]=='+') || (stack[j]=='-'))
                        while((stack[j]=='+')||(stack[j]=='-'))
                        {
                            printf("%c ",stack[j]);
                            j--;
                            if(j==-1)
                                break;
                        }
                t=j;
                t++;
                stack[t]=a[i][0];
            }
            else
if(((stack[t]=='-')||(stack[t]=='+'))&&((a[i][0]=='-')||(a[i][0]=='+')))
            {
                printf("%c ",stack[t]);
                t--;
                t++;
                stack[t]=a[i][0];
            }
            else if(a[i][0]==')')
            {
                j=t;
                while(stack[j]!='(')
                {
                    printf("%c ",stack[j]);
                    j--;
                }
                t=j;
                t--;
            }
```

```
        }
        else
            printf("%s ",a[i]);
        i++;
    }
    printf(" ");
    while(t!=-1)
    {
        printf("%c ",stack[t]);
        t--;
    }
    printf("\n\n");
    return 0;
}
int main()      //driver function
{
    int i,j,x;
    printf("First Create a Stack. | Enter Size of stack to be created.\n");
    scanf("%d",&n);
    st=(int*)malloc(n*sizeof(int));
    printf("Stack of size %d has been created. | Now use the functions 
below.\n",n);
    while(1)        //infinite loop for user interaction
    {

printf("\n\n----------------------------------------------------------\n");
        printf("ENTER 1: To push() element into stack\n");
        printf("ENTER 2: To pop() element out of stack\n");
        printf("ENTER 3: To implement headrecursion() and tailrecursion()\n");
        printf("ENTER 4: To convert infix expression to postfix.| 
infix2postfix()\n");
        printf("ENTER 5: To print() the status of STACK\n");
        printf("ENTER 6: To exit the program\n");

printf("\n----------------------------------------------------------\n\n");
        scanf("%d",&x);
        switch(x)
        {
            case 1: push(); break;
            case 2: pop(); break;
            case 3: printf("Sum of natural numbers till n will be implemented 
by tail and head recursion\n");
                    printf("Please enter value of n\n");
                    scanf("%d",&i);
                    printf("Result through head recursion:\n");
                    printf("%d\n",sumhead(i));
                    printf("\n\nResult through tail recursion:\n");
                    printf("=%d\n",sumtail(i));
                    break;
            case 4: infix2postfix(); break;
            case 5: print(); break;
            case 6: return 0; break;
            default: printf("Wrong input\n"); break;
        }
    }
}
```

Output:

```
First Create a Stack. | Enter Size of stack to be created.
3
Stack of size 3 has been created. | Now use the functions below.


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

1
Enter the element you want to push
1


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------
```

```
1
Enter the element you want to push
2


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

1
Enter the element you want to push
3


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

1
Overflow of stack.


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program
```

```
----------------------------------------------------------

5


Current status of stack:
1  2  3



----------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

----------------------------------------------------------

2
3 is popped out of stack



----------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

----------------------------------------------------------

2
2 is popped out of stack



----------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

2
1 is popped out of stack


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

2
Underflow of stack


---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

3
Sum of natural numbers till n will be implemented by tail and head recursion
Please enter value of n
7
Result through head recursion:
+7+6+5+4+3+2+1=28


Result through tail recursion:
1+2+3+4+5+6+7=28
```

```
---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

4
Enter an infix expression (USE $ TO TERMINATE):
a + b - ( c + d ) / 2 - 98 % 7 $
a b + c d + 2 / - 98 7  % -



---------------------------------------------------------
ENTER 1: To push() element into stack
ENTER 2: To pop() element out of stack
ENTER 3: To implement headrecursion() and tailrecursion()
ENTER 4: To convert infix expression to postfix.| infix2postfix()
ENTER 5: To print() the status of STACK
ENTER 6: To exit the program

---------------------------------------------------------

6
========================================================================
```

Week 7

Problem Statement:

Create Function for each question. And call it using switch case.(eg: case 1 : Q1) .
Presentation matters.
1. Implement SortStack() using recursion.
2. Print a function(name of your choice) for printing a pattern without using any loop.
3. Implement Infix2Prefix() function which will convert an infix expression to prefix.
4. Implement stack's operation Push() using linked list.
Call of the every functions will be given into the Print() function.

Description: *The Programs aims at implementing sortstack , recursion, infix to prefix, and stack using linked list.*
STACK: Expression evaluation and syntax parsing. Calculators employing reverse Polish notation use a ***stack*** *structure to hold values. Expressions can be represented in prefix, postfix or infix notations and conversion from one form to another may be accomplished using a* ***stack****.*
RECURSION: ***Recursion*** *is a programming technique in which a method makes a call to itself to solve a particular problem. Such methods are called* ***recursive****.* ***Recursion*** *is a programming technique whose correct usage leads to elegant solutions to certain problems like Tower Of Hanoi etc.*
NOTATION (PREFIX): In prefix the operator comes before both the operand. Prefix notation is nearly as easy to process; it's used in Lisp

Solution:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node         //self referencial structure
{
      int data;
      struct node *link;
};
```

```
int ssort()     //function to sort a stack
{
     int n=1,i,m,k=0,x,swap;
     struct node *start,*ptr,*temp,*temp2;
     start=NULL;
     printf("Enter the size of stack to be created\n");
     scanf("%d",&m);
     while(1)
     {
          printf("\nEnter 1 to push()\n");
          printf("Enter 2 to print current status of stack\n");
          printf("Enter 3 to exit stack function\n");
          printf("Enter 4 to sort the stack at current\n");
          scanf("%d",&n);
          if(n==3)
               return 0;
          switch(n)
          {
               case 1: printf("Enter data to push into stack\n");
                         scanf("%d",&x);
                         if(k>=m)
                         {
                              printf("WARNING!!!: Stack Overflow\n");
                              break;
                         }
                         else if(k==0)
                         {
                              ptr=(struct node*)malloc(sizeof(struct node));
                              start=ptr;
                              start->data=x;
                              start->link=NULL;
                              temp=start;
                              k++;
                         }
                         else
                         {
                              ptr=(struct node*)malloc(sizeof(struct node));
                              ptr->data=x;
                              ptr->link=temp;
                              temp=ptr;
                              k++;
                         }
```

```
                        break;
                case 2:    ptr=temp;
                   printf("CURRENT STACK:  ");
                        while(ptr!=NULL)
                        {
                             printf("%d ",ptr->data);
                             ptr=ptr->link;
                        }
                        printf("\n");
                        break;
                case 4: ptr =temp;
                        temp2=temp;
                        while(ptr!=NULL)
                        {
                             temp2=temp;
                             while(temp2->link!=NULL)
                             {
                                  if(temp2->data>temp2->link->data)
                                  {
                                       swap=temp2->data;
                                       temp2->data=temp2->link->data;
                                       temp2->link->data=swap;
                                  }
                                  temp2=temp2->link;

                             }
                             ptr=ptr->link;
                        }
                        printf("STACK is sorted. You may check by printing
the current stack.\n");
                        break;
                default:
                             printf("WARNING: Wrong Input\n");
                             break;
            }

     }
     return 0;
}
int llist()     //function to implement stack in linked list
{
     int n=1,i,m,k=0,x;
     struct node *start,*ptr,*temp,*temp2;
```

```
start=NULL;
printf("Enter the size of stack to be created\n");
scanf("%d",&m);
while(1)
{
     printf("\nEnter 1 to push()\n");
     printf("Enter 2 to print current status of stack\n");
     printf("Enter 3 to exit stack function\n");
     scanf("%d",&n);
     if(n==3)
          return 0;
     switch(n)
     {
          case 1: printf("Enter data to push into stack\n");
                    scanf("%d",&x);
                    if(k>=m)
                    {
                         printf("WARNING!!!: Stack Overflow\n");
                         break;
                    }
                    else if(k==0)
                    {
                         ptr=(struct node*)malloc(sizeof(struct node));
                         start=ptr;
                         start->data=x;
                         start->link=NULL;
                         temp=start;
                         k++;
                    }
                    else
                    {
                         ptr=(struct node*)malloc(sizeof(struct node));
                         ptr->data=x;
                         ptr->link=temp;
                         temp=ptr;
                         k++;
                    }
                    break;
          case 2:   ptr=temp;
                    while(ptr!=NULL)
                    {
                         printf("%d ",ptr->data);
                         ptr=ptr->link;
```

```
                              }
                              printf("\n");
                              break;
                    default:
                                   printf("WARNING!!!: Wrong Input\n");
                                   break;
               }

          }
          return 0;
}
int recur(int k)          //function to recurse
{
          if(k==0)
               return 0;
          else
          {
               printf("*");
               recur(k-1);
          }
          return 0;
}

int pattern(int n)        //function to print pattern using recursion
{
          if(n==0)
               return 0;
          else
          {
               recur(n);
               printf("\n");
               pattern(n-1);
          }
          return 0;
}
int infix2prefix()        //function to convert infix to prefix
{
    char a[1000][10],stack[1000],c[1000][10],d,e[1000][10],temp[100];
    int b[1000],i,j=0,p=0,len,t=-1,k=0;
    printf("Enter an infix expression (USE $ TO TERMINATE):\n");
    for(i=0; ;i++)
    {
        scanf("%s",a[i]);
```

```
            if(a[i][0]=='$')
                break;
        }
        a[i][0]='\0';
        len=i;
         for(i=0;i<len;i++)
              strcpy(c[i],a[len-i-1]);
         c[len][0]='\0';
         for(i=0;i<len;i++)
         {
              if(c[i][0]=='(')
                   c[i][0]=')';
              else if(c[i][0]==')')
                   c[i][0]='(';
         }
        i=0;
        while(c[i][0]!='\0')        //various cases for infix to prefix
        {

    if((c[i][0]=='*')||(c[i][0]=='/')||(c[i][0]=='%')||(c[i][0]=='+')||(c[i][0]=='
    -')||(c[i][0]=='(')||(c[i][0]==')')||(c[i][0]=='^'))
            {
                if(t==-1)
                {
                    t++;
                    stack[t]=c[i][0];
                }
                else if(((stack[t]=='(')||(c[i][0]=='('))&&(c[i][0]!=')'))
                {
                    t++;
                    stack[t]=c[i][0];
                }
                else if((stack[t]=='^')&&(c[i][0]=='^'))
                {
                          e[k][0]=stack[t];
                          e[k++][1]='\0';

                    t--;
                    t++;
                    stack[t]=c[i][0];
                }
                else if((c[i][0]=='^')&&(stack[t]!='^'))
                {
                    t++;
                    stack[t]=c[i][0];
                }
                else if((stack[t]=='^')&&(c[i][0]!=')'))
                {

                          e[k][0]=stack[t];
                          e[k++][1]='\0';
                    t--;
                    if(t>-1)
                    if(((c[i][0]=='-')||(c[i][0]=='+'))&&(stack[t]!='('))
                    {

                               e[k][0]=stack[t];
                               e[k++][1]='\0';
                        t--;
                        if(t>-1)
                            if((stack[t]=='-')||(stack[t]=='+'))
                            {

                                    e[k][0]=stack[t];
                                    e[k++][1]='\0';
                            t--;
                            }
                    }
                    if(t>-1)

if(((c[i][0]=='/')||(c[i][0]=='*')||(c[i][0]=='%'))&&((stack[t]=='/')||(stack[
t]=='*')||(stack[t]=='%')))
                    {

                               e[k][0]=stack[t];
                               e[k++][1]='\0';
                        t--;
                    }
                    t++;
                    stack[t]=c[i][0];
                }
                else
if(((stack[t]=='-')||(stack[t]=='+'))&&((c[i][0]=='%')||(c[i][0]=='*')||(c[i][
0]=='/')||(c[i][0]=='^')))
                {
                    t++;
```

```
            stack[t]=c[i][0];
        }
        else
if(((stack[t]=='%')||(stack[t]=='/')||(stack[t]=='*')||(c[i][0]=='^'))&&(c[i][
0]!=')')&&(c[i][0]!='^'))
        {

                  e[k][0]=stack[t];
                  e[k++][1]='\0';
            t--;
            j=t;
            if((c[i][0]=='+') || (c[i][0]=='-'))
                if((stack[j]=='+') || (stack[j]=='-'))
                    while((stack[j]=='+')||(stack[j]=='-'))
                    {

                                   e[k][0]=stack[j];
                                   e[k++][1]='\0';
                        j--;
                        if(j==-1)
                            break;
                    }
            t=j;
            t++;
            stack[t]=c[i][0];
        }
        else
if(((stack[t]=='-')||(stack[t]=='+'))&&((c[i][0]=='-')||(c[i][0]=='+')))
        {

                  e[k][0]=stack[t];
                  e[k++][1]='\0';
            t--;
            t++;
            stack[t]=c[i][0];
        }
        else if(c[i][0]==')')
        {
            j=t;
            while(stack[j]!='(')
            {

                       e[k][0]=stack[j];
```

```
                              e[k++][1]='\0';
                       j--;
                   }
                   t=j;
                   t--;
               }
           }
           else
                   strcpy(e[k++],c[i]);

           i++;
       }
       printf(" ");
       while(t!=-1)
       {
             e[k][0]=stack[t];
             e[k++][1]='\0';

           t--;
       }
        for(i=k-1;i>=0;i--)
             printf("%s ",e[i]);          //print the prefix
       printf("\n\n");
       return 0;
   }
   int main()      //driver function
   {
        int x,n,i,j;
        while(1)
        {

   printf("___________________________________________________________\n");
             printf("Enter 1 for sortstack()\n");
             printf("Enter 2 for printing a pattern without using loop\n");
             printf("Enter 3 for Inorder() Traversal\n");
             printf("Enter 4 for infix to prefix conversion\n");
             printf("Enter 5 for PUSH() using linked list\n");
             printf("Enter 6 to exit() program\n");

   printf("___________________________________________________________\n");
             scanf("%d",&x);
             if(x==6)
                  return 0;
```

```
          switch(x)
          {
               case 1: ssort();
                         break;
               case 2: printf("Enter an integer\n");
                         scanf("%d",&n);
                         pattern(n);
                         break;
               case 5: llist();
                         break;
               case 4: infix2prefix();
                         break;
               default: printf("WARNING!!!: Invalid Input\n");
                         break;

          }
     }

}
```

Output:

```
Enter 1 for sortstack()
Enter 2 for printing a pattern without using loo
Enter 3 for Inorder() Traversal
Enter 4 for infix to prefix conversion
Enter 5 for PUSH() using linked list
Enter 6 to exit() program
___________________________________________________________
1
Enter the size of stack to be created
5

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
3

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
1

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
2

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
5

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
4

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
1
Enter data to push into stack
7
WARNING!!!: Stack Overflow

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
2
CURRENT STACK:  4 5 2 1 3

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
4
STACK is sorted. You may check by printing the current stack.

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
2
CURRENT STACK:  1 2 3 4 5

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
Enter 4 to sort the stack at current
3
___________________________________________________________
Enter 1 for sortstack()
Enter 2 for printing a pattern without using loop
Enter 3 for Inorder() Traversal
Enter 4 for infix to prefix conversion
Enter 5 for PUSH() using linked list
Enter 6 to exit() program
___________________________________________________________
2
Enter an integer
9
*********
********
*******
******
*****
****
***
**
*
```

```
___________________________________________________________
Enter 1 for sortstack()
Enter 2 for printing a pattern without using loop
Enter 3 for Inorder() Traversal
Enter 4 for infix to prefix conversion
Enter 5 for PUSH() using linked list
Enter 6 to exit() program
___________________________________________________________
4
Enter an infix expression (USE $ TO TERMINATE):
a / b + c - ( d - e ) % 10 $
 + / a b - c % - d e 10


___________________________________________________________
Enter 1 for sortstack()
Enter 2 for printing a pattern without using loop
Enter 3 for Inorder() Traversal
Enter 4 for infix to prefix conversion
Enter 5 for PUSH() using linked list
Enter 6 to exit() program
___________________________________________________________
5
Enter the size of stack to be created
6

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
1

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
2

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
```

```
Enter data to push into stack
3

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
4

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
5

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
6

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
1
Enter data to push into stack
7
WARNING!!!: Stack Overflow

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
2
6 5 4 3 2 1

Enter 1 to push()
Enter 2 to print current status of stack
Enter 3 to exit stack function
3
```
____

```
Enter 1 for sortstack()
Enter 2 for printing a pattern without using loop
Enter 3 for Inorder() Traversal
Enter 4 for infix to prefix conversion
Enter 5 for PUSH() using linked list
Enter 6 to exit() program
____________________________________________________________
6
```

===

Week 8

Problem Statement:

Create Function for each question. And call it using switch case.(eg: case 1: Q1) .
Presentation matters.

1. *Implement enqueue() function using array which will insert values in the queue.*
2. *Implement dequeue() function using linked list which will delete values from queue.*
3. *Implement Isfull() function, which will inform the user if queue is full.*
4. *Implement Isempty() function, which will inform the user if queue is empty.*
5. *Implement Print(), in which call of every function will be given.*

Description: *The Programs aims at implementation of queue.*

*QUEUE: **Queue** is used when things don't have to be processed immediately, but have to be processed in **F**irst **I**n **F**irst **O**ut order like **Breadth First Search**. This property of Queue makes it also useful in following kind of scenarios.*

***1)** When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.*

***2)** When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.*

Solution:

```
#include<stdio.h>
#include<stdlib.h>
struct node     //self referencial structure
{
    int data;
    struct node *next;
};
int arrisfull(int rear,int x,int det)       //to check if queue is full
{
    if(rear==x-1)
    {
        printf("\tQueue is Full\n");
        return 0;
    }
    else if(det==0)
    {
        if(rear!=-2)
            printf("\tQueue is not Full\n");
        else
            printf("\tQueue is not Full. But also it cannot accomodate more as
LIMIT has REACHED!\n");
        return 1;
    }
    return 1;
}

int arrisempty(int front,int rear,int det)      //to check if queue is empty
{
    if(front<0 || front>rear)
    {
        printf("\tQueue is empty\n");
        return 0;
    }
    else if(det==0)
    {
        printf("\tQueue is not empty\n");
        return 1;
    }
    return 1;
}

int arrenq(int *rear,int *front,int *ptr,int x,int y)       //insertion in
queue
{
    if(arrisfull(*rear,x,1)==0)
        return 0;
    if(*rear==-2)
    {
        printf("\tCannot insert more. LIMIT REACHED\n");
        return 0;
    }
    *rear+=1;
    ptr[*rear]=y;
    if(*front==-1)
        *front=0;
}

int arrdeq(int *rear,int *front,int *ptr,int x)     //to delete from queue
{
    if(arrisempty(*front,*rear,1)==0)
        return 0;
    printf("\t%d is deleted\n",ptr[*front]);
    *front+=1;
    if(*front==x)
        *rear=-2;
}

int qarray(int x)       //implementation of queue in array
{
    int i,j,n,y,*ptr,rear=-1,front=-1;
    ptr=(int*)malloc(x*sizeof(int));
    while(1)
    {
        printf("\n_______________________________________________\n");
        printf("\tENTER 1 : For insertion in queue\n");
        printf("\tENTER 2 : For deletion from queue\n");
        printf("\tENTER 3 : For calling Isempty()\n");
        printf("\tENTER 4 : For calling Isfull()\n");
        printf("\tENTER 5 : To jump to MAIN MENU\n");
        printf("\n_______________________________________________\n\n");
        printf("\tEnter your choice-->  ");
        scanf("%d",&n);
        if(n==5)
            return 0;
        switch(n)
        {
            case 1 : printf("\tEnter the data to be inserted-->  ");
                     scanf("%d",&y);
                     arrenq(&rear,&front,ptr,x,y);
                     break;
            case 2 : arrdeq(&rear,&front,ptr,x);
                     break;
            case 3 : arrisempty(front,rear,0);
                     break;
            case 4 : arrisfull(rear,x,0);
                     break;
            default : printf("\tWrong Input\n");
                      break;
        }
    }
}
struct node *ptr,*head,*rear,*front,*temp;
int flag=0;

int listisfull(int ri,int x)    //to check if queue list is full
{
    if(ri==x-1)
    {
        printf("\tQueue is Full\n");
        return 0;
    }
    else
    {
        if(ri!=-2)
            printf("\tQueue is not Full\n");
        else
            printf("\tQueue is not Full. But also it cannot accomodate more as
LIMIT has REACHED!\n");
        return 1;
    }
    return 1;
}

int listisempty(int fi,int ri)      //to check if queue list is empty
```

```
{
    if(fi<0 || fi>ri)
    {
        printf("\tQueue is empty\n");
        return 0;
    }
    else
    {
        printf("\tQueue is not empty\n");
        return 1;
    }
    return 1;
}
int listenq(int *ri,int *fi,int x,int y)        //insertion in queue list
{
    if(*ri==x-1)
    {
        printf("\tQueue is Full\n");
        return 0;
    }
    if(*ri==-2)
    {
        printf("\tCannot insert more. LIMIT REACHED\n");
        return 0;
    }
    if(rear==NULL)
    {
        rear=ptr;
        rear->data=y;
        rear->next=NULL;
        *ri+=1;
        *fi+=1;
        front=rear;
    }
    else if(rear!=NULL)
    {
        ptr=(struct node*)malloc(sizeof(struct node));
        rear->next=ptr;
        rear=ptr;
        rear->data=y;
        rear->next=NULL;
        *ri+=1;
        if(flag==1)
        {
            temp=rear;
            flag=0;
        }
    }
}
int listdeq(int *ri,int *fi,int x)      //deletion from queue list
{
    struct node *ptr1;
    if(*fi<0 || *fi>*ri)
    {
        printf("\tQueue is empty\n");
        return 0;
    }
    if(front==NULL)
    {
        front=temp;
    }
    printf("\t%d is deleted\n",front->data);
    ptr1=front;
    if(front!=rear)
        front=front->next;
    else
    {
        flag=1;
        front=NULL;
    }

    *fi+=1;
    ptr1->next=NULL;
    if(*fi==x)
    {
        *ri=-2;
    }
}

int qlist(int x)        //implementation of queue using linked list
{
    int i,j,n,y,ri=-1,fi=-1;

    ptr=(struct node*)malloc(sizeof(struct node));
    head=ptr;
    rear=front=NULL;
    while(1)
    {
        printf("\n_______________________________________________\n");
        printf("\tENTER 1 : For insertion in queue\n");
        printf("\tENTER 2 : For deletion from queue\n");
        printf("\tENTER 3 : For calling Isempty()\n");
        printf("\tENTER 4 : For calling Isfull()\n");
        printf("\tENTER 5 : To jump to MAIN MENU\n");
        printf("\n_______________________________________________\n\n");
        printf("\tEnter your choice-->  ");
        scanf("%d",&n);
        if(n==5)
            return 0;
        switch(n)
        {
            case 1 : printf("\tEnter the data to be inserted-->  ");
                     scanf("%d",&y);
                     listenq(&ri,&fi,x,y);
                     //printf("%d\n",rear->data);
                     break;
            case 2 : listdeq(&ri,&fi,x);
                     break;
            case 3 : listisempty(fi,ri);
                     break;
            case 4 : listisfull(ri,x);
                     break;
            default : printf("\tWrong Input\n");
                      break;
        }
    }
}

int main()          //driver function
{
    int n,x,i,j;
    while(1)
    {
        printf("\n_______________________________________________\n");
        printf("ENTER 1 : For implementing queue using array\n");
        printf("ENTER 2 : For implementing queue using linked list\n");
        printf("ENTER 3 : To terminate the program\n");
        printf("[Isfull() and Isempty() are implemented inside 1 & 2]\n\n");
        printf("\n_______________________________________________\n");
        printf("Your Choice-->  ");
        scanf("%d",&n);
        if(n==3)
            break;
        switch(n)
        {
            case 1 : printf("Enter the size of Queue to implement:\n");
                     scanf("%d",&x);
                     qarray(x);
                     break;
            case 2 : printf("Enter the size of Queue to implement:\n");
                     scanf("%d",&x);
                     qlist(x);
                     break;
            default : printf("Wrong Input\n");
                      break;
        }

    }
}
```

Output:
[Using linked list]

```
_______________________________________________
ENTER 1 : For implementing queue using array
ENTER 2 : For implementing queue using linked list
ENTER 3 : To terminate the program
[Isfull() and Isempty() are implemented inside 1 & 2]



_______________________________________________
Your Choice-->  2
Enter the size of Queue to implement:
5


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU
```

```
_______________________________________________

        Enter your choice-->  3
        Queue is empty


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  4
        Queue is not Full


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  2
        Queue is empty


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  1


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  2

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  3

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  4

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________
```

```
        Enter your choice-->  1
        Enter the data to be inserted-->  5

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  6
        Queue is Full

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  3
        Queue is not empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  4
        Queue is Full

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        1 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        2 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        3 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        4 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        5 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  3
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  4
        Queue is not Full. But also it cannot accomodate more as LIMIT has
REACH
ED!

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  5

_______________________________________________
ENTER 1 : For implementing queue using array
ENTER 2 : For implementing queue using linked list
ENTER 3 : To terminate the program
[Isfull() and Isempty() are implemented inside 1 & 2]


_______________________________________________
Your Choice-->  3
```

[Using Array]

```
_______________________________________________
ENTER 1 : For implementing queue using array
ENTER 2 : For implementing queue using linked list
ENTER 3 : To terminate the program
[Isfull() and Isempty() are implemented inside 1 & 2]


_______________________________________________
Your Choice-->  1
Enter the size of Queue to implement:
5
```

```
_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  3
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  4
        Queue is not Full

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU
```

```
_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  1


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  2


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  3


_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU


_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  4


_______________________________________________
```

```
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  5

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  1
        Enter the data to be inserted-->  6
        Queue is Full

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  3
        Queue is not empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU
```

```
_______________________________________________

        Enter your choice-->  4
        Queue is Full

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        1 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        2 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        3 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        4 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        5 is deleted

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  2
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________
```

```
        Enter your choice-->  3
        Queue is empty

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  4
        Queue is not Full. But also it cannot accomodate more as LIMIT has
REACH
ED!

_______________________________________________
        ENTER 1 : For insertion in queue
        ENTER 2 : For deletion from queue
        ENTER 3 : For calling Isempty()
        ENTER 4 : For calling Isfull()
        ENTER 5 : To jump to MAIN MENU

_______________________________________________

        Enter your choice-->  5

_______________________________________________
ENTER 1 : For implementing queue using array
ENTER 2 : For implementing queue using linked list
ENTER 3 : To terminate the program
[Isfull() and Isempty() are implemented inside 1 & 2]

_____________________________________________
Your Choice-->  3
==========================================================================
```

MID TERM Evaluation Problem

Problem Statement: *Given an array of size 'n' and an integer 'k', first divide the array into two halves and then print both the subarrays after reversing the first 'k' values of each subarray. One should not use extra array to do this task.*

Solution:

```
#include<stdio.h>

int main()
{
      FILE *fp;
      int n,i,a[1000],k,mid,temp;
      fp=fopen("input.txt","r");       //input from file
      printf("Enter size of array: ");
      scanf("%d",&n);
      for(i=0;i<n;i++)
            fscanf(fp,"%d",&a[i]);
      printf("Given array:\n");
      for(i=0;i<n;i++)
            printf("%d ",a[i]);
      printf("\n");
      lab:
      printf("Enter value of k: ");
      scanf("%d",&k);
      if(n%2==0)
            mid=n/2;
      else
            mid=(n+1)/2;
      if(k>mid && n%2==0)         //error handling
      {
            printf("ERROR!!! Value of k cant exceed %d in this case\nAgain
Enter value of k\n",mid);
            goto lab;
      }
      if(k>mid-1 && n%2==1)       //error handling for odd case
      {
            printf("ERROR!!! Value of k cant exceed %d in this case\nAgain
Enter value of k\n",mid);
            goto lab;
      }
      for(i=0;i<k/2;i++)
      {
            temp=a[i];
            a[i]=a[k-i-1];
            a[k-i-1]=temp;
        }
        for(i=0;i<0+k/2;i++)
        {
            temp=a[mid+i];
            a[mid+i]=a[mid+k-i-1];
            a[mid+k-i-1]=temp;
        }
        printf("First Subarray:\n");
        for(i=0;i<mid;i++)
            printf("%d ",a[i]);
        printf("\nSecond Subarray:\n");
        for(i=mid;i<n;i++)
            printf("%d ",a[i]);
        printf("\n");
        fclose(fp);           //closing file

}
```

Output:

```
Enter size of array: 12
Given array:
12 13 14 15 16 17 18 19 20 21 22 23
Enter value of k: 6
First Subarray:
17 16 15 14 13 12
Second Subarray:
23 22 21 20 19 18
```

WEEK 9

Problem Statement: *1. Implement preorder_traversal() function using linked list which will traverse a tree in root first, left node second and right node in the last sequence.*

2. Implement inorder_traversal() function using linked list which will traverse a tree in left first, root node second and right node in the last, sequence.

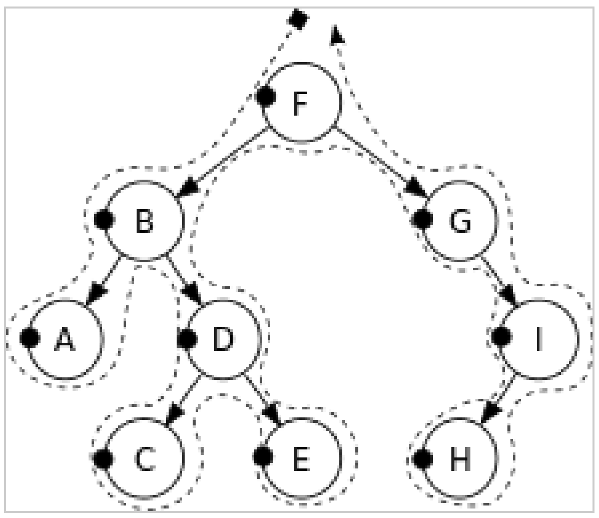
3. Implement postorder_traversal() function, using linked list which will traverse a tree in left first, root node second and right node in the last sequence.

4. In a classroom there are N students sitting in the class, and M students are standing out of the class. When everyone students enters he has given K(1,2,3..) cards. When a student enters to the class he wishes to be seated with that student which has same number of cards. Help the student getting his exact seat and denote it with "Yes" or "No"

Input	*Output*
1,2,3,7,8	*NO*
1,2,3,7,8,3	*Yes*

Description: *This program focuses on the ways of doing depth traversals on trees, which are namely Preorder, Postorder and Inorder. These traversals are used in different purposes, like if a parse tree of a formula is given then preorder traversal of that parse tree will give prefix expression and postorder traversal will give postfix notation of that formula. This program also describes solving a real life class/student problem using linked lists and algorithms for deletion, insertion of nodes in a singly linked list.*

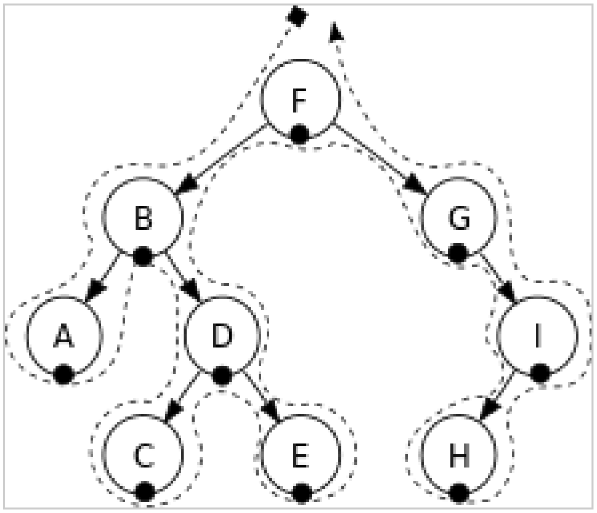
***Pre-order**:*



Pre-order: F, B, A, D, C, E, G, I, H.

1. *Check if the current node is empty / null*
2. *Display the data part of the root (or current node).*
3. *Traverse the left subtree by recursively calling the pre-order function.*
4. *Traverse the right subtree by recursively calling the pre-order function.*

In-order:

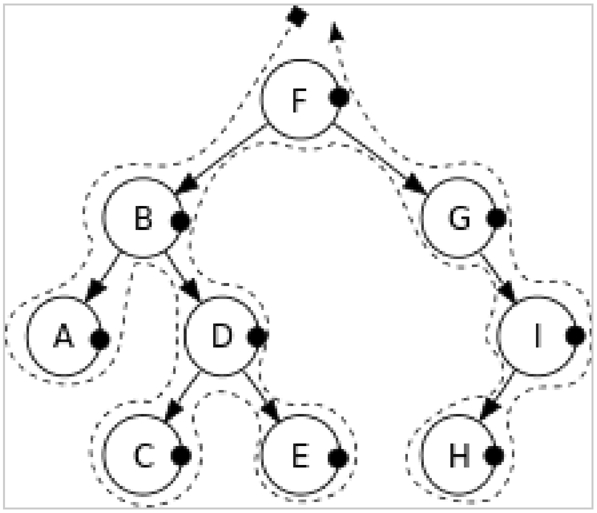


In-order: A, B, C, D, E, F, G, H, I.

1. *Check if the current node is empty / null*
2. *Traverse the left subtree by recursively calling the in-order function.*
3. *Display the data part of the root (or current node).*
4. *Traverse the right subtree by recursively calling the in-order function.*

In a search tree, in-order traversal retrieves data in sorted order.

Post-order:



Post-order: A, C, E, D, B, H, I, G, F.

1. *Check if the current node is empty / null*

2. *Traverse the left subtree by recursively calling the post-order function.*
3. *Traverse the right subtree by recursively calling the post-order function.*
4. *Display the data part of the root (or current node).*

Solution:

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

struct node //node for tree
{
     int data;
     struct node* left;
     struct node* right;
};
struct node1  //node for class-student problem
{
      int data;
      struct node *link;
};

/* Helper function that allocates a new node */
struct node* newNode(int data)
{
     struct node* node = (struct node*)malloc(sizeof(struct node));
     node->data = data;
     node->left = NULL;
     node->right = NULL;

     return(node);
}
/* Given a binary tree, print postorder traversal. */
void printPostorder(struct node* node)
{
     if (node == NULL)
        return;
     printPostorder(node->left);
     printPostorder(node->right);
     printf("%d ", node->data);
}
/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
     if (node == NULL)
```

```
          return;
     printInorder(node->left);
     printf("%d ", node->data);
     printInorder(node->right);
}
/* Given a binary tree, print its nodes in preorder*/
void printPreorder(struct node* node)
{
     if (node == NULL)
          return;
     printf("%d ", node->data);
     printPreorder(node->left);
     printPreorder(node->right);
}

int main()
{
    int n=1,i,m,k,in,out,tt,flag,x,a[10];
    struct node1 *start,*ptr,*temp,*temp2,*start2,*ptr2,*temp3,*curr;
    printf("First Enter 7 elements to build a tree:\n");
    for(i=0;i<7;i++)
        scanf("%d",&a[i]);
    struct node *root  = newNode(a[0]);
     root->left             = newNode(a[1]);
     root->right           = newNode(a[2]);
     root->left->left     = newNode(a[3]);
     root->left->right   = newNode(a[4]);
     root->right->left=newNode(a[5]);
     root->right->right=newNode(a[6]);
    while(1)
    {
        printf("Enter 1 for pre-order traversal\n");
        printf("Enter 2 for in-order traversal\n");
        printf("Enter 3 for post-order traversal\n");
        printf("Enter 4 for the class/student problem\n");
        printf("Enter 0 to exit the program\n");
        scanf("%d",&x);
        if(x==0)
            break;
        switch(x)
        {
            case 1: printf("\t\t   %d\n\n\t\t%d\t%d\n\n\t   %d\t   %d  %d\t
%d\n",root->data,root->left->data,root->right->data,root->left->left->data,roo
t->left->right->data,root->right->left->data,root->right->right->data);
                    printf("Pre-order traversal: ");
                    printPreorder(root);
                    printf("\n");
                    break;
            case 2: printf("\t\t   %d\n\n\t\t%d\t%d\n\n\t   %d\t   %d  %d\t
%d\n",root->data,root->left->data,root->right->data,root->left->left->data,roo
t->left->right->data,root->right->left->data,root->right->right->data);
                    printf("In-order traversal: ");
                    printInorder(root);
                    printf("\n");
                    break;
            case 3: printf("\t\t   %d\n\n\t\t%d\t%d\n\n\t   %d\t   %d  %d\t
%d\n",root->data,root->left->data,root->right->data,root->left->left->data,roo
t->left->right->data,root->right->left->data,root->right->right->data);
                    printf("Post-order traversal: ");
                    printPostorder(root);
                    printf("\n");
                    break;
            case 4: start=NULL;
                    printf("Enter the no. of students standing out\n");
                    scanf("%d",&out);
                    printf("Enter the no. of students sitting inside
class\n");
                    scanf("%d",&in);
                    printf("Enter the %d cards which are inside the class as a
list\n",in);
                    m=in;
                    for(i=0;i<m;i++)
                    {
                        if(i==0)
                        {
                            ptr=(struct node1*)malloc(sizeof(struct node1));
                            //printf("Enter the data for 1st node: \n");
                            scanf("%d",&ptr->data);
                            ptr->link=NULL;
                            start=ptr;
                            //printf("1st node constructed\n");
                        }
                        else if(i>0)
                        {
```

```
                        ptr=(struct node1*)malloc(sizeof(struct node1));
                        //printf("Enter the data for %d node: \n",i+1);
                        scanf("%d",&ptr->data);
                        ptr->link=NULL;
                        if(i==1)
                            start->link=ptr;
                        else
                            temp->link=ptr;
                        temp=ptr;
                    }

                }
                curr=ptr;
                printf("Enter the %d cards which are outside the class the
one by one\n",out);
                m=out;
                for(i=0;i<m;i++)
                {
                    scanf("%d",&tt);
                    ptr2=start;
                    flag=0;
                    while(ptr2!=NULL)
                    {
                        if(tt==ptr2->data)
                        {
                            printf("YES\n");
                            flag=1;
                                temp2=ptr2->link;
                                ptr=(struct node1*)malloc(sizeof(struct
node1));

                                ptr->data=tt;
                                ptr->link=temp2;
                                ptr2->link=ptr;
                                break;
                        }
                        ptr2=ptr2->link;
                    }
                    if(flag==0)
                    {
                        printf("No\n");
                        ptr=(struct node1*)malloc(sizeof(struct node1));
                        ptr->data=tt;
                        ptr->link=NULL;
```

```
                            curr->link=ptr;
                            curr=ptr;
                        }
                    }
                    printf("After Entering in the class the configuration
becomes\n");
                    ptr=start;
                    while(ptr!=NULL)
                    {
                        printf("%d ",ptr->data);
                        ptr=ptr->link;
                    }
                    printf("\n");
                    break;
            default: printf("Wrong Choice\n");
        }
    }
    return 0;
}
```

Output:

```
First Enter 7 elements to build a tree:
1 2 3 4 5 6 7
Enter 1 for pre-order traversal
Enter 2 for in-order traversal
Enter 3 for post-order traversal
Enter 4 for the class/student problem
Enter 0 to exit the program
1
                   1

                2       3

           4       5  6    7
Pre-order traversal: 1 2 4 5 3 6 7
Enter 1 for pre-order traversal
Enter 2 for in-order traversal
Enter 3 for post-order traversal
Enter 4 for the class/student problem
Enter 0 to exit the program
2
                   1
                2       3

           4       5   6    7
In-order traversal: 4 2 5 1 6 3 7
Enter 1 for pre-order traversal
Enter 2 for in-order traversal
Enter 3 for post-order traversal
Enter 4 for the class/student problem
Enter 0 to exit the program
3
                   1

                2       3

           4       5   6    7
Post-order traversal: 4 5 2 6 7 3 1
Enter 1 for pre-order traversal
Enter 2 for in-order traversal
Enter 3 for post-order traversal
Enter 4 for the class/student problem
Enter 0 to exit the program
4
Enter the no. of students standing out
6
Enter the no. of students sitting inside class
4
Enter the 4 cards which are inside the class as a list

1 2 3 4
Enter the 6 cards which are outside the class the one by one
1
YES
5
No
5
YES
6
No
6
YES
2
YES
After Entering in the class the configuration becomes
```

```
1 1 2 2 3 4 5 6 6
Enter 1 for pre-order traversal
Enter 2 for in-order traversal
Enter 3 for post-order traversal
Enter 4 for the class/student problem
Enter 0 to exit the program
0
```

==