List of Experiments

1a. Write a program to find the solution for the maximum subsequence sum problem using
   I. Linear Algorithm
b. Write a program to implement the following searching techniques
   a. Linear search
   b. Binary search.
2. Write a program for the following sorting techniques
   a. Insertion
   b. Shell
3. Write a program for the following sorting techniques
   a. Merge sort.
   b. Quick sort.
4. Write a program to implement stack and queues using arrays
5 a. Write a program to convert an infix to postfix expression
   b. Write a program to evaluate a postfix expression
6. Write a program to implement all operations on SLL
7. Write a program to implement all operations on DLL.
8a. Write a program to implement stack using linked list
   b. Write a program to implement queue using linked list
9. Write a program to implement the following polynomial Operations.
   a. Addition
   b. Subtraction
10. Write a program to create a binary tree & do the following operation on it.
   I. Insertion
   II. Deletion
   III. Traversal
11. Write a non-recursive program for in order tree traversals.
12. Write a program to implement separate chaining Hashing Technique
13. Write a program to implement two stacks using single array
14. Write a program to multiply 2 polynomials

KONERU LAKSHMAIAH UNIVERSITY

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

2/4,3/4,4/4 I Semester COMPUTER CENTER TIME-TABLE FOR THE A.Y.2011-2012

with effect from: 07-07-11

	1	2		3	4		5	6		7	8
DAY	**9.00am To 9.50am**	**9.50am To 10.40am**	**10.40am To 11.00am**	**11.00am To 11.50am**	**11.50am To 12.40pm**	**12.40pm To 1.30pm**	**1.30pm To 2.20pm**	**2.20pm To 3.10pm**	**3.10pm To 3.20pm**	**3.20pm To 4.10pm**	**4.10pm To 5.00pm**
MON	**SS LAB (II/IV SEC-A)**		**B**				**L**		**B**	**DBMS LAB (III/IV SEC-A)**	
TUE			**R**	**IP LAB (IV/IV SEC-A)**			**U**		**R**	**DS LAB (II/IV SEC-B)**	
WED			**E**	**DS LAB (II/IV SEC-A)**			**N**		**E**	**DBMS LAB (III/IV SEC-B)**	
THU	**SS LAB (II/IV SEC-B)**		**A**				**C**		**A**		
FRI			**K**	**IP LAB (IV/IV SEC- B)**			**H**		**K**		

(U.JYOTHI KAMESWARI)
(Dr.S.BALAJI)
LAB-I/C **HOD**
ECM

Evaluation procedure:

The student will be evaluated on continuous day to day evaluation basis, internal and external lab exams. Day to day evaluation marks will be given based on the students performance in the lab – whether he is able to complete the experiment with in the lab class time and based on viva voice.
Internal and external lab exams consists of different components like write up, execution viva and record. Apart from these components, student will be allotted 5 marks for the record basing on whether he is showing the record for every experiment in the immediate week and 5 marks will be allotted for the attendance.

Evaluation Scheme:

EC NO.	Evaluation Component	Weightage(marks)
1	Day to day evaluation ,viva voice	10
2	Record	5
3	Internal exam	20
4	Attendance	5
6	Comprehensive Examination	60

Attendance Scale:

% of Attendance	Marks awarded
≥ 95	5
≥ 90 and < 95	4
≥ 85 and < 90	3
≥ 80 and < 85	2
≥ 75 and < 80	1

EXERCISE NO : 1

1a. Write a program to find the solution for the maximum subsequence sum problem using

I. Linear Algorithm

Aim:

A program to find the solution for the maximum subsequence sum problem using Linear Algorithm

Source Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20],i,j,n,sum;
int alg4(int[],int);
clrscr();
printf("enter n value\n");
scanf("%d",&n);
printf("enter array values\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
sum=alg4(a,n);
printf("sum of linear alg is: %d",sum);
}
int alg4(int a[20],int n)
{
int thissum,maxsum,j;
maxsum=0;
thissum=0;
for(j=0;j<n;j++)
{
thissum += a[j];
if(thissum > maxsum)
maxsum=thissum;
else if(thissum<0)
thissum=0;
}
return maxsum;
}
```

Output:

Enter n value 6
Enter array values -2 11 -4 13 -5 -2
Sum of linear alg is : 20

1b. Write a program to implement the following searching techniques

a. Linear search
b. Binary search.

Aim:

A Program to Implement Linear & Binary Search Techniques

Source Code:

```
#include<stdio.h>
#include<conio.h>
void lsearch(int a[],int n,int ele);
void bsearch(int a[],int n,int ele);
void ssort(int a[],int n);
void read(int a[],int n);
void main()
{
int a[50],n,op,ele;
clrscr();
printf("\nMENU");
printf("\n1.LINEAR SEARCH");
printf("\n2. BINARY SEARCH");
printf("\n3.EXIT");
printf("\nEnter your option:");
scanf("%d",&op);
if(op>0&&op<3)
{
printf("\nEnter the size of array:");
scanf("%d",&n);
read(a,n);
printf("\nEnter the elements to search:");
scanf("%d",&ele);
}
switch(op)
{
case 1:
        lsearch(a,n,ele);
        break;
case 2:
        bsearch(a,n,ele);
        break;
case 3:exit(0);
default:printf("\nINVALID OPTION!!!");
}
getch();
}
void read(int a[],int n)
{
int i;
printf("\nEnter the %d elements below\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
}
void ssort(int a[],int n)
{
int i,j,temp;
for(i=0;i<n-1;i++)
```

```
{
for(j=i+1;j<n;j++)
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
void lsearch(int a[],int n,int ele)
{
int flag=0,i;
for(i=0;i<n;i++)
{
if(a[i]==ele)
{
flag=1;
break;
}
}
if(flag==1)
printf("\nElement in the list");
else
printf("\nElement not in the list");
}
void bsearch(int a[],int n,int ele)
{
int ll=0,ul=n-1,mid;
ssort(a,n);
while(ll<=ul)
{
mid=(ll+ul)/2;
if(a[mid]==ele)
{
printf("\nElement found");
break;
}
else if(a[mid]>ele)
ul=mid-1;
else if(a[mid]<ele)
ll=mid+1;
 }
 if(ul>ll)
printf("\nElement is not in the list");
}
```

Result:

Linear & Binary Searching techniques implemented & tested with sample data.

EXERCISE NO : 2

Write a program for the following sorting techniques

a. Insertion
b. Shell

Aim:

A Program to Implement Insertion and Shell Sorting Techniques

Source Code:

```
#include<stdio.h>
#include<conio.h>
void print(int a[],int n);
void insertion(int a[],int n);
void shell(int a[],int n);
void main()
  {
   int n,a[30],op,i;
   char ch;
   while(1)
    {
     clrscr();
     printf("\nMENU\n1.INSERTION SORT\n2.SHELL SORT\n3.EXIT\n");
     printf("\nENTER YOUR CHOICE:  ");
     scanf("%d",&op);
      if(op >0 && op<3)
       {
        printf("\n\n Enter the number of elements:");
        scanf("%d",&n);
        printf("\n\n Enter the elements:");
          for(i=0;i<n;++i)
                scanf("%d",&a[i]);
       }
      switch(op)
       {
        case 1:insertion(a,n);print(a,n);
                  break;
        case 2:shell(a,n);print(a,n);
                  break;
        case 3: exit(0);
        default:printf("\n invalid choice!");
        }
        printf("\n do you wish to continue(Y/N):  ");
         flushall();
         ch=getchar();
         if(ch=='n'|| ch=='N')
          break;
         }
       getch();
      }
```

```
void print(int a[],int n)
{
int i;
printf("\n the sorted list is : ");
for(i=0;i<n;i++)
{
printf("\n\t %d ",a[i]);
}
}

void insertion(int a[],int n)
  {
    int j,p;
int temp;
for(p=1;p<n;p++)
{
temp=a[p];
for(j=p;j>0 && a[j-1] > temp;j--)
a[j]=a[j-1];
a[j]=temp;
 }
}

void shell(int a[],int n)
  {
   int i,j,increment,temp;
for(increment = n/2;increment > 0; increment/=2)
for(i=increment;i<n;i++)
{
Temp=a[i];
For(j=i;j>=increment;j-=increment)
If(tmp<a[j-increment])
a[j]=a[j-increment];
Else
Break;
a[j]=temp;
}

   }
```

Result:

Insertion and Shell Sorting Techniques were implemented & tested with sample data.

EXERCISE NO : 3

Write a program for the following sorting techniques

a. Merge sort.
b. Quick sort.

Aim:

A Program to Implement Merge & Quick Sorting Techniques

Source Code:

Merge Sort:

```
#include<stdio.h>
void mergesort(int a[20],int );
msort(int a[20],int [],int ,int);
merge(int a[20],int [],int ,int ,int);
main()
{
int a[20],n,i;
clrscr();
printf("enter size of array");
scanf("%d",&n);
printf("enter values");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
mergesort(a,n);
printf("the sorted list is \n");
for(i=0;i<n;i++)
printf("%d\n",a[i]);
}
void mergesort(int a[20],int n)
{
int *temparray;
temparray=(int*) malloc(n*sizeof(int));
if(temparray!=NULL)
{
msort(a,temparray,0,n-1);
free(temparray);
}
}
msort(int a[20],int temparray[20],int left,int right)
{
int center;
if(left<right)
{
center=(left+right)/2;
msort(a,temparray,left,center);
msort(a,temparray,center+1,right);
merge(a,temparray,left,center+1,right);
}
}
merge(int a[20],int temparray[20],int lpos,int rpos,int rightend)
{
int i,temppos, noe,leftend;
temppos=lpos;
leftend=rpos-1;
noe=rightend-lpos+1;
```

```
while(lpos<=leftend && rpos<=rightend)
{
if(a[lpos]<a[rpos])
temparray[temppos++]=a[lpos++];
else
temparray[temppos++]=a[rpos++];
}
while(lpos<=leftend)
temparray[temppos++]=a[lpos++];
while(rpos<=rightend)
temparray[temppos++]=a[rpos++];
for(i=0;i<noe;i++,rightend--)
a[rightend]=temparray[rightend];

}
```

Quick Sort:

```
#include<stdio.h>
void main()
{
int  a[], n,i;
printf("enter no. of elements");
scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,n);
}
Void  qsort(int a[], int left, int right)

{

int i,j,pivot;

Pivot = median3(a,left,right);
i=left;j=right-1;
for(;;)

{while(a[++i] <pivot){}

  While(a[--j] > pivot){}

If(i<j)

Swap(a[i],a[j]);

Else

Break;

}

Swap(a[i],a[right-1]);

Qsort(a,left,i-1);

Qsort(a,i+1,right);

}

Void quicksort(int a[i],int n)

{

Qsort(a,0,n-1);
```

```
}


int median3(int a[],int left,int right)
{
Int center;
Center = (left + right)/2;
If(a[left] > a[center])
Swap(&a[left],&a[center]);
If(a[left] > a[right])
Swap(&a[left],&a[right]);
If(a[center] > a[right])
Swap(&a[center],&a[right]);
Swap(&a[center],&a[right-1]);
Return(a[right-1];
}
```

Result:

Merge & Quick Sorting Techniques were implemented & tested with sample data.

EXERCISE NO : 4

Write a program to implement stack and queues using arrays.

Aim:

A Program to Implement & Manipulate Stack and Queue Data Structures.

Source Code:

Stack:

```
#include<stdio.h>
#include<conio.h>
#define size 5
int s[22],i;
int tos=-1;
void push(int);
int pop();
void display();
void main()
{
int ch,ele;
clrscr();
do
{
printf("\nMENU");
printf("\n1.TO PUSH ELEMENT");
printf("\n2.TO POP ELEMENT");
printf("\n3.TO DISPLAY ELEMENT");
printf("\n4.EXIT");
printf("\nEnter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:
        printf("\nEnter element to push");
        scanf("%d",&ele);
        push(ele);
        break;
case 2:
        ele=pop();
        printf("\nthe poped element is %d",ele);
        break;
case 3:
        display();
        break;
case 4:
        exit(0);
default:printf("\nWrong option!!");
}
}
while(1);
}
void push(int x)
{ /* inserting an element into the stack */
if(tos>= n-1)
```

```
printf(“Overflow on push”);
else
s[++tos] = x;
}
int pop()
{ /* deleting an element from the stack */
int y;
if(tos < = 1)
{
printf(“ Underflow on pop”);
return(-1);
}
else
{
y = s[tos];
tos--;
return(y);
}
}
 void display()
        {
        int i;
        if(tos==NULL)
        printf("\nStack is empty");
        else
        {
        for(i=0;i<tos;i++)
        printf("\n%d\t",s[i]);
        }
        }
```

Queue:

```
#include<stdio.h>
#include<conio.h>
#define size 7
int a[22],i,front=-1,rear=-1;
void insert(int ele);
int delete();
void display();
void main()
{
int op,ele,i;
clrscr();
do{
printf("\nMENU");
printf("\n1.INSERTION");
printf("\n2.DELETION");
printf("\n3.DISPLAY");
printf("\n4.EXIT");
printf("\nEnter your option:");
scanf("%d",&op);
switch(op)
{
case 1:
```

```
            printf("\nEnter the element to be insert:");
            scanf("%d",&ele);
            insert(ele);
            printf("\nElement inserted");
            printf("\nElements after insertion are");
            for(i=0;i<rear;i++)
            {
            printf("\n%d",a[i]);
            }
            }           break;
      case 2:
            ele=delete();
            printf("\nElements after deletion are");
            for(i=front;i<rear;i++)
            {
            printf("\n%d",a[i]);
            }
            }break;
      case 3:
            display();
            break;
      case 4: exit(0);
      default:printf("\nInvalid Choice");
      }
      }
      while(1);
      }
void insert(int x)
{ /* inserts an element into the queue */
if(rear >= n-1)
{
printf( “overflow on insert”);
return;
}
else
{
a[++rear] = x;
if(front = = -1)
front ++;
}
}
int delete()
{ /* deletes front element from the queue */
int y;
if( front = = -1)
{
printf( “overflow on delete”);
return(-1);
}
else
{
y = a[front];
if( front = = rear )
front = rear = -1;
```

```
else
front++;
return(y); } }
void display()
      {
      int i;
      if(rear==0)
      printf("\nQUEUE EMPTY");
      else
      {
      for(i=front;i<rear;i++)
      {
      printf("%d\t",a[i]);
      }
      }
```

Result:

Stack and Queue Data Structures are implemented & Operations over the data structures are verified with test data.

EXERCISE NO : 5

5a. Write a program to convert an infix to postfix expression
b. Write a program to evaluate a postfix expression

Aim:

A Program to write a program that converts an infix to postfix expression & to evaluate it.

Source Code:

a. program to convert an infix to postfix expression

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define n 10
int issymbol(char x);
void infixtopost(char ine[],char pe[]);
int isopen(char x);
int isoperator(char x);
int precedence(char x);
char top(char[] );
int pop(char s[]);
void push(char s[],int x);
int tos;
char ine[50],pe[50];
void main()
{
 int i=0;
 clrscr();
 printf("enter infix expression:");
 gets(ine);

 while((ine[i]!='\0'))
 {
    infixtopost(ine,pe);
    ++i;
 }
printf("postfix expression is\n");
for(i=0;pe[i]!='\0';++i)
    printf("%c",pe[i]);
getch();
}

void infixtopost(char ine[],char pe[])
{
   int i,j,px,py;
   char x,y,s[20];
   tos=-1;
   i=j=0;
   while(ine[i]!='\0')
   {
     x=ine[i];
     if(!isoperator(x)&&!issymbol(x))

            pe[j++]=x;
```

```
 else if(isoperator(x))
 {
          if(tos!=-1)
          {
            y=top(s);
            if(!issymbol(y))
                    {
                      py=precedence(y);
                      px=precedence(x);
                    while(tos!=-1&&px<=py)
                    {
                      pe[j++]=pop(s);
                      y=pop(s);
                      py=precedence(y);
                    }
              }
          }
             push(s,x);
          }
            else
            {
                    if(isopen(x))
                       push(s,x);
                    else
                      {
                        while(top(s)!='(')
                                pe[j++]=pop(s);
                                pop(s);
                      }
            }
            ++i;
  }
}
int issymbol(char x)
{
  if((x=='(')||(x==')'))
          return(1);
  else
           return(0);
}
int isopen(char x)
{
 if(x=='(')
         return(1);
 else
          return(0);
}
int isoperator(char x)
{
        if(x=='*'||x=='+'||x=='-'||x=='/'||x=='%')
          return(1);
  else
           return(0);
}
int precedence(char x)
{
  if(x=='^')
         return(3);
  else if(x=='%'||x=='*'||x=='/')
```

```
                return(2);
              else
                        return(0);
  }
  char top(char s[])
  {
    if(tos>-1)
            return(s[tos]);
    else
             return(-1);
  }
  int pop(char s[])
  {
    int y;
    if(tos<=-1)
           printf("under flow\n");
    else
    {
            y=s[tos];
            tos--;
    }
    return(y);
  }
  void push(char s[],int x)
  {
    if(tos>=n-1)
          printf("over flow\n");
    else
    {
            tos++;
            s[tos]=x;
    }     return;}
```

b. A program to evaluate the postfix expression.

```
#include<stdio.h>
#include<conio.h>
int tos,n;
int postfixvalue(char pe[]);
int isoperator(char x);
int pop(int s[50]);
void push(int s[50],int);
void main()
{
  char pe[50];
  int i=0,p;
  clrscr();
  printf("enter the post fix expression:\n");
  while(1)
  {
    scanf("%c",&pe[i]);
    if(pe[i]=='#')
            break;
            ++i;
  }
  p=postfixvalue(pe);
  printf("The value of post fix expression is %d\n",p);
  getch();
```

```
}
int postfixvalue(char pe[])
{
  int i,x,y,z,s[50];
  tos=-1;
  i=0;
  while(pe[i]!='#')
  {
     if(isoperator(pe[i]))
     {
                y=pop(s);
                x=pop(s);
                switch(pe[i])
                {
                   case '*': z=x*y;
                             break;
                   case '+':z=x+y;
                             break;
                   case '-':z=x-y;
                             break;
                   case '/':z=x/y;
                             break;
                   case '%':z=x%y;
                             break;
                }
                push(s,z);
    }
    else
    {
              printf("enter the value of %c:",pe[i]);
              scanf("%d",&x);
              push(s,x);
    }
    ++i;
    }
      return(s[tos]);
    }

void push(int s[],int x)
{
   if(tos>=50)
      printf("over flow\n");
   else
   {
       tos++;
       s[tos]=x;
   }
  return;
}
int pop(int s[])
{
  int y;
  if(tos<=-1)
   {
      printf("stack is empty:\n");
      return(-1);
   }
 else
 {
```

```
    y=s[tos];
    tos--;
}
   return(y);
}

int isoperator(char x)
{
  if(x=='*'||x=='%'||x=='/'||x=='+'||x=='-')
     return(1);
  else
            return(0);
 }
```

Result:

program implemented for conversion of an infix expression to postfix expression & to evaluate it & is tested with sample data.

EXERCISE NO : 6

Write a program to implement all operations on SLL

Aim:

A Program to Implement all operations like Insertion, deletion, sorting, merging reversing on SLL.

Source Code:

```
#include<stdio.h>
#include<conio.h>
struct node
{
 int data;
 struct node *next;
};
typedef struct node node;
node *head1,*temp,*head2,*head3,*newnode;
void create(node *);
void display(node *);
node *insertion(node *);
node *Delete(node *);
node *merge();
node *sort(node *);
void main()
{
 int choice;
 char ch;
 while(1)
 {
   clrscr();

printf("\n\t\t\tMENU\n\t\t1.create\n\t\t2.insertion\n\t\t3.deletion\n\t\t4.merging\n\t\t5.sorting\n\t\t6.Rev
erse\n\t7.exit");
    printf("\n\t\tenter any choice: ");
    scanf("%d",&choice);
     if(choice>0&&choice<7)
       head1=(node *)malloc(sizeof(node));
    switch(choice)
    {
      case 1:  create(head1);
                 display(head1);
                 break;
      case 2:   create(head1);
                  head1=insertion(head1);
                  display(head1);
                  break;
      case 3:   create(head1);
                  head1=Delete(head1);
                  display(head1);
                  break;
      case 4:   create(head1);
                  head2=(node *)malloc(sizeof(node));
                  printf("\n enter data values of other list:\n");
                  create(head2);
                  head3=merge();
```

```
                    display(head3);
                    break;
     case 5:
                    create(head1);
                    head1=sort(head1);
                    display(head1);
                    break;
     case 6: create(head1);
                reverse(head1);
                break;
     case 7:    exit(0);
     default: printf("\n\t Invalid choice! \n");
 }
  printf("\n do you wish to continue(Y/N): ");
  flushall();
  ch=getchar();
  if(ch=='N'||ch=='n')
    break;
 }
 getch();

 }

 void create(node *head)
 {
  printf("enter data(Enter 0 to stop):");
  scanf("%d",&head->data);
  if(head->data==0)
  {
     head->next=NULL;
     return;
  }
  else
  {
      head->next=(node *)malloc(sizeof(node));
      create(head->next);
    }
 }

void display(node *head)
{
  if(head->next==NULL)
  {

    return;
   }
  else
    {  if(head->data==0)
         return;
      printf("\n\t%d",head->data);
      display(head->next);
    }
}

 node * insertion(node *head)
{
  int pos,choice,count=0;
  printf(" specify where the insertion is to be made: ");
  printf("\n\t1.beginning\n\t2.any position\n\t3.end");
```

```
  printf("\n enter your specification: ");
  scanf("%d",&choice);
  if(choice>0&&choice<4)
   {
     newnode=(node *)malloc(sizeof(node));
     printf("\n enter data to be inserted: ");
     scanf("%d",&newnode->data);
   }
  if(choice==1)
  {
     newnode->next=head;
     head=newnode;

  }
  else if(choice==2)
        {
          printf("\n enter the position of insertion : ");
          scanf("%d", &pos);
          temp=head;
          while(count<pos-2)
          {
                  temp=temp->next;
                  count++;
          }
                newnode->next=temp->next;
                temp->next=newnode;

        }
       else if(choice==3)
                  {
                    temp=head;
                    while(temp->next!=NULL)
                            {
                              temp=temp->next;
                            }
                              temp->next=newnode;
                              newnode->next=NULL;

                   }
  return(head);
}
node * Delete(struct node *head)
{
  int pos,choice,count =0;
  printf(" specify where the deletion is to be made:\n\t1.begining\n\t2.any position\n\t3.end");
  printf("\n enter your specification: ");
  scanf("%d",&choice);
  if(choice==1)
  {
    head=head->next;

  }
  else if(choice==2)
       {
          printf("\n enter position of deletion: ");
          scanf("%d",&pos);
          temp=head;
          while(count<pos-2)
                  {
```

```
                temp=temp->next;
                count++;
               }
              temp->next=temp->next->next;

      }
     else if(choice==3)
              {
                temp=head;
                while(temp->next->next!=NULL)
                        temp=temp->next;
                      temp->next=NULL;

              }
   return(head);
}

node *merge(void)
{
  node *temp1,*temp2,*temp3;
  temp1=head1;
  temp2=head2;
  temp3=(node *)malloc(sizeof(node));
  head3=temp3;
  while((temp1!=NULL)&&(temp2!=NULL))
     {
       if(temp1->data<temp2->data)
              {
                 temp3->data=temp1->data;
                 temp1=temp1->next;
              }
        else
              {
                 temp3->data=temp2->data;
                 temp2=temp2->next;
              }
       temp3->next=(node *)malloc(sizeof(node));
       temp3=temp3->next;
     }
      if(temp1!=NULL)
              temp3->next=temp2;
      else if(temp2!=NULL)
              temp3->next=temp1;


      temp3->data=0;
      temp3->next=NULL;
     return(head3);
}
node * sort(node *head)
{
  int p;
   node *t1,*t2;
  t1=head;
  for(;t1!=NULL;t1=t1->next)
    for(t2=t1->next;t2!=NULL;t2=t2->next)
      {
        if(t1->data<t2->data)
              {
```

```
                        p=t1->data;
                        t1->data=t2->data;
                        t2->data=p;
                       }
                }
            return(head);
        }
reverse(list l)
{
list curr,next,prev=NULL;
curr=l->next;
while(curr)
{
l->next=curr;
next=curr->next;
curr->next=prev;
prev=curr;
curr=next;
}
return;
}
```

Result:

Operations on single linked list were performed & tested with sample data.

EXERCISE NO : 7

Write a program to implement all operations on DLL

Aim:

A Program to Implement all operations like Insertion, deletion, sorting, merging reversing on DLL

Source Code:

```
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
struct node *prev;
struct node *next;
}*head1,*head2,*temp,*head,*temp1,*temp2,*newnode;
void create();
void insert(int n);
void Delete(int n);
void sort();
void merge();
void reverse();
void search();
void display();
void main()
{
int n,op;
clrscr();
do
{
printf("\nMENU");
printf("\n1.insertion");
printf("\n2.deletion");
printf("\n3.sorting");
printf("\n4.merging");
printf("\n5.reversing");
printf("\n6.searching");
printf("\n7.exit");
printf("\nenter your option");
scanf("%d",&op);
if(op>0&&op<7)
{
printf("\nenter elements in list");
create();
head1=head;
}
switch(op)
{
case 1:
     insert(n);
     display();
     break;
case 2:
      Delete(n);
      display();
      break;
case 3:
```

```
        sort();
        display();
        break;
case 4:
        merge();
        display();
        break;
case 5:
        reverse();

        break;
case 6:
        search();
        break;
case 7:
       exit(0);
default:
        printf("\ninvalid option");
}
}while(1);
}
void create()
{
int x,n=1;
head=(struct node *)malloc(sizeof(struct node));
printf("\nenter data");
scanf("%d",&x);
head->data=x;
head->next=NULL;
temp=head;
while(x!=0)
{
newnode=(struct node *)malloc(sizeof(struct node));
printf("\nenter data");
scanf("%d",&x);
if(x!=0)
{
newnode->prev=temp;
newnode->data=x;
newnode->next=NULL;
temp->next=newnode;
temp=newnode;
++n;
}
else
break;
}
}
void insert(int n)
{
int pos,x,i;
printf("\nenter position to insert");
scanf("%d",&pos);
printf("\nenter element to insert");
scanf("%d",&x);
newnode=(struct node *)malloc(sizeof(struct node));
if(pos==1)
{
newnode->data=x;
```

```
newnode->prev=NULL;
newnode->next=head;
head=newnode;
}
else if(pos==n-1)
{
temp=head;
while(temp->next!=NULL)
temp=temp->next;
newnode->data=x;
newnode->next=NULL;
newnode->prev=temp;
temp->next=newnode;
temp=newnode;
}
else
{
temp=head;
for(i=1;i<pos-1;++i)
temp=temp->next;
newnode->data=x;
newnode->next=temp->next;
temp->next->prev=newnode;
temp->next=newnode;
newnode->prev=temp;
}
}
void Delete(int n)
{
int pos,i;
printf("\nenter position to delete");
scanf("%d",&pos);
if(pos==1)
{
head=head->next;
head->prev=NULL;
}
else if(pos==n-1)
{
temp=head;
while(temp->next!=NULL)
temp=temp->next;
temp->prev->next=NULL;
}
else
{
temp=head;
for(i=1;i<pos;++i)
temp=temp->next;
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
}
}
void sort()
{
int t;
for(temp1=head;temp1!=NULL;temp1=temp1->next)
for(temp2=head;temp2->next!=NULL;temp2=temp2->next)
{
```

```
if((temp1->data)<(temp2->data))
{
t=temp1->data;
temp1->data=temp2->data;
temp2->data=t;
}
}
}
  node *merge(void)
  {
    node *temp1,*temp2,*temp3;
    temp1=head1;
    temp2=head2;
    temp3=(node *)malloc(sizeof(node));
    head3=temp3;
    while((temp1!=NULL)&&(temp2!=NULL))
       {
         if(temp1->data<temp2->data)
                {
                   temp3->data=temp1->data;
                   temp1=temp1->next;
                }
          else
                 {
                    temp3->data=temp2->data;
                    temp2=temp2->next;
                 }
         temp3->next=(node *)malloc(sizeof(node));
         temp3=temp3->next;
       }
         if(temp1!=NULL)
                temp3->next=temp2;
         else if(temp2!=NULL)
                  temp3->next=temp1;


         temp3->data=0;
         temp3->next=NULL;
        return(head3);
 }

void reverse()
{
}
void search()
{
int ele,flag=0;
printf("\nenter element to search");
scanf("%d",&ele);
for(temp=head;temp!=NULL;temp=temp->next)
{
if((temp->data)==ele)
{
flag=1;
break;
}
}
if(flag==1)
printf("\n%d is in list",ele);
```

```
else
printf("\n%d is not in list",ele);
}
void display()
{
temp=head;
printf("\nelements in list are:");
while(temp!=NULL)
{
printf("\n%d",temp->data);
temp=temp->next;
}
}
```

Result:

Operations on Double linked list were performed & tested with sample data.

EXERCISE NO : 8

STACKS & QUEUES

Write a Program to Implement Stack & Queues using Linked List.

Aim:

A Program to Implement Stack & Queues using Linked List.

Source Code:

```
#include<stdio.h>
#include<conio.h>
struct stack
{
int data;
struct stack *next;
}*tos;

void create(int n);
void push(int n);
int pop();
void display();
void main()
{
int op,n,t;
tos=NULL;
clrscr();
do
```

```
{
printf("\n MENU\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT");
printf("\n ENter Your Option:");
scanf("%d",&op);
switch(op)
{
case 1:printf("\n Enter Data:");
scanf("%d",&n);
if(tos==NULL)
{
create(n);
display();
}
else
{
push(n);
display();
}
break;
case 2:if(tos==NULL)
{
printf("\n Stack Is Empty");
}
else
{
t=pop();
printf("\n Poped Element is %d",t);
display();
break;
}
case 3:
display();
break;
case 4:exit(0);
default:printf("\n Invalid Option");
}
}
while(1);
}

void create(int n)
{
tos=(struct stack *)malloc(sizeof(struct stack));
tos->data=n;
tos->next=NULL;
}

void push(int n)
{
struct stack *nn;
nn=(struct stack *)malloc(sizeof(struct stack));
nn->data=n;
nn->next=tos;
tos=nn;
}

int pop()
{
int t;
```

```
t=tos->data;
tos=tos->next;
return(t);
}


void display()
{
struct stack *p;
if(tos==NULL)
{
printf("\n Stack Is Empty");
}
else
{
printf("\n Elements In stack are:");
p=tos;
while(p!=NULL)
{
printf("\t-> %d",p->data);
p=p->next; }}}
```

Result:

Stacks & Queue Data structures were implemented using Linked Lists & also all possible operations were tested over them with sample data.

EXERCISE NO : 9

Write a Program to Implement Polynomial Data Structure using linked lists.

Aim:

A Program to Implement Polynomial Data Structure using linked lists.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct term
{
        float coef;
        int xp;
        struct term *next;
};
typedef struct term *tptr;
tptr createheader();
tptr findpos(tptr p,int e);
tptr findprev(tptr p,int e);
tptr find(tptr p,int e);
void insertterm(tptr p,int e,float c);
void deleteterm(tptr p,int e);
tptr sumpoly(tptr p1,tptr p2);
tptr subpoly(tptr p1,tptr p2);
void display(tptr p);
void main()
{
        tptr p,p2,res;
```

```
int e,op;
float c;
char ch;
p=createheader();
clrscr();
printf("\nEnter the coefeicient,xpower:");
while(1)
{
        scanf("%f%d",&c,&e);
        insertterm(p,e,c);
        if(e==0)
        break;
}
display(p);
while(1)
{
        printf("\n\n\tMENU");
        printf("\n\n1.FIND");
        printf("\n\n2.INSERT");
        printf("\n\n3.DELETE");
        printf("\n\n4.ADDITION");
        printf("\n\n 5. SUBTRACTION");
        printf("\n\n6.EXIT");
        printf("\nEnter your option:");
        scanf("%d",&op);
        switch(op)
        {
        case 1:
                printf("\nEnter the exponent to be found");
                scanf("%d",&e);
                res=find(p,e);
                if(res==NULL)
                printf("\nThe power doesnot exist");
                else
                {
                        printf("%f* x^%d",res->next->coef,e);
                }
                break;
        case 2:
                printf("\nEnter the coefficient and xpower");
                scanf("%f%d",&c,&e);
                insertterm(p,e,c);
                display(p);
                break;
        case 3:
                printf("\nEnter the xpower to be deleted");
                scanf("%d",&e);
                deleteterm(p,e);
                display(p);
                break;
        case 4:
                p2=createheader();
                printf("\nEnter the terms in the second poly");
                while(1)
                {
                        scanf("%f%d",&c,&e);
                        insertterm(p2,e,c);
                        if(e==0)
                        break;
```

```
                        }
                        res=sumpoly(p,p2);
                        display(res);
                        break;
                case 5: p2=createheader();
                        printf("\nEnter the terms in the second poly");
                        while(1)
                        {
                                scanf("%f%d",&c,&e);
                                insertterm(p2,e,c);
                                if(e==0)
                                break;
                        }
                        res=subpoly(p,p2);
                        display(res);
                        break;
                case 6:
                        exit(0);
                default:printf("\nInvalid choice");
                }
                printf("\nDo you wish to continue(y/n)");
                flushall();
                scanf("%c",&ch);
                if(ch=='n'||ch=='N')
                break;
        }
}
tptr createheader()
{
        tptr p;
        p=(tptr)malloc(sizeof(struct term));
        if(p!=NULL)
        p->next=NULL;
        else
        {
        printf("\nOut ofspace");
        exit(0);
        }
        return(p);
}
tptr find(tptr p,int e)
{
        tptr temp;
        temp=p;
        while(temp!=NULL){
        if(temp->next->xp==e)
        return(temp);
        temp=temp->next;
        }
        return(NULL);
}
tptr findpos(tptr p,int e)
{
        tptr temp;
        temp=p;
        while((temp->next!=NULL)&&(temp->next->xp>e))
        temp=temp->next;
        return(temp);
}
```

```
void insertterm(tptr p,int e,float c)
{
	tptr pos,temp;
	if(p->next==NULL)
	{
		temp=(tptr)malloc(sizeof(struct term));
		temp->coef=c;
		temp->xp=e;
		p->next=temp;
		temp->next=NULL;
	}
	else
	{
	pos=find(p,e);
	if(pos!=NULL)
	pos->next->coef+=c;
	else
	{
	  temp=(tptr)malloc(sizeof(struct term));
	  temp->coef=c;
	  temp->xp=e;
	  pos=findpos(p,e);
	  temp->next=pos->next;
	  pos->next=temp;
	}
	}
}
void deleteterm(tptr p,int e)
{
	tptr pos,temp;
	pos=find(p,e);
	if(pos!=NULL)
	{
		temp=pos->next;
		pos->next=pos->next->next;
		free(temp);
	}
	else
	printf("\nThe power not found");
}
tptr sumpoly(tptr p1,tptr p2)
{
	tptr t1,t2,p3;
	float c;
	p3=createheader();
	t1=p1->next;
	t2=p2->next;
	while((t1!=NULL)&&(t2!=NULL))
	{
		if(t1->xp<t2->xp)
		{
			insertterm(p3,t2->xp,t2->coef);
			t2=t2->next;
		}
		else if(t1->xp>t2->xp)
		{
			insertterm(p3,t1->xp,t1->coef);
			t1=t1->next;
		}
```

```
                else
                {
                        c=t1->coef+t2->coef;
                        insertterm(p3,t1->xp,c);
                        t1=t1->next;
                        t2=t2->next;
                }
        }
        while(t1!=NULL)
        {
                insertterm(p3,t1->xp,t1->coef);
                t1=t1->next;
        }
        while(t2!=NULL)
        {
                insertterm(p3,t2->xp,t2->coef);
                t2=t2->next;
        }
        return(p3);
}

tptr subpoly(tptr p1,tptr p2)
{
        tptr t1,t2,p3;
        float c;
        p3=createheader();
        t1=p1->next;
        t2=p2->next;
        while((t1!=NULL)&&(t2!=NULL))
        {
                if(t1->xp<t2->xp)
                {
                        insertterm(p3,t2->xp,t2->coef);
                        t2=t2->next;
                }
                else if(t1->xp>t2->xp)
                {
                        insertterm(p3,t1->xp,t1->coef);
                        t1=t1->next;
                }
                else
                {
                        c=t1->coef-t2->coef;
                        insertterm(p3,t1->xp,c);
                        t1=t1->next;
                        t2=t2->next;
                }
        }
        while(t1!=NULL)
        {
                insertterm(p3,t1->xp,t1->coef);
                t1=t1->next;
        }
        while(t2!=NULL)
        {
                insertterm(p3,t2->xp,t2->coef);
                t2=t2->next;
        }
        return(p3);
```

```
}
void display(tptr p)
{
        tptr temp;
        temp=p->next;
        while(temp!=NULL)
        {
        printf("%+f*x^%d",temp->coef,temp->xp);
        temp=temp->next;
        }
}
```

Result:

Polynomial Data Structure using linked lists were implemented & their operations were tested with sample data.

EXERCISE NO : 10

Write a program to create a binary search tree & do the following operation on it.

I. Insertion
II. Deletion
III. Traversal

Aim:

A Program to Implement binary search tree & to manipulate it.

Source Code:

```
#include<stdio.h>
#include<malloc.h>
struct treenode;
typedef struct treenode* position;
typedef struct treenode* tree;
struct treenode
{
int element;
tree left;
tree right;
};
position findmin(tree);
position insert(int x, tree t)
{
if(t==NULL)
{
t=(tree)malloc(sizeof(struct treenode));
if(t==NULL)
{
```

```
printf("OUT OF SPACE");
return;
}
else
  {
t->element=x;
t->left=NULL;
t->right=NULL;
}
}
else
if(x<t->element)
t->left=insert(x,t->left);
else
if(x>t->element)
t->right=insert(x,t->right);
return(t);
}
tree delete(int x, tree t)
{
position tempcell;
if(t==NULL)
printf("Not Found");
else
if(x<t->element)
t->left=delete(x,t->left);
else
  if(x>t->element)
t->right=delete(x,t->right);
else
{
if(t->left && t->right)
{
tempcell=findmin(t->right);
t->element=tempcell->element;
t->right=delete(t->element,t->right);
}
else
{
tempcell=t;
if(t->left==NULL)
t=t->right;
else
if(t->right==NULL)
t=t->left;
free(tempcell);
}
}
return t;
}
 position findmin(tree t)
{
tree temp;
temp=t;
if(temp!=NULL)
while(temp->left!=NULL)
temp=temp->left;
return(temp);
}
```

```
inorder(tree t)
{
if(t!=NULL)
{
inorder(t->left);
printf("%d\t",t->element);
inorder(t->right);
}
}


preorder(tree t)
{
if(t!=NULL)
 {
printf("%d\t",t->element);
preorder(t->left);
preorder(t->right);
}
}
postorder(tree t)
{
if(t!=NULL)
{
postorder(t->left);
postorder(t->right);
printf("%d\t",t->element);
}
}
main()
{
tree t=NULL;
int ch=0,x;
while(ch!=6)
{
printf("1.insert\n2.delete\n3.inorder\n4.preorder\n");
printf("5.postorder\n6.exit\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("enter element to insert\n");
scanf("%d",&x);
t=insert(x,t);
break;
case 2:
printf("enter element to delete");
scanf("%d",&x);
delete(x,t);
break;
case 3:
printf("\nthe inoder is :\n");
inorder(t);
break;
case 4:
printf("\nthe preoder is :\n");
preorder(t);
```

```
break;
case 5:
printf("\nthe postoder is :\n");
postorder(t);
break;
}
}
}
```

Result:

Binary Search Tree is implemented & tested with sample data.

EXERCISE NO : 11

Write a non-recursive program for in order tree traversals.

Aim:

A Program to Implement non recursive tree traversals over a BST

Source Code:

```
#include<conio.h>

#include<stdio.h>

typedef struct treenode *trptr;

struct treenode

{

trptr lptr;

int data;

trptr rptr;

};

int tos;

void push(trptr s[],trptr);
```

```
trptr pop(trptr s[]);

void recpreorder(trptr);

trptr insertbst(trptr,int);

void recpostorder(trptr);

void recinorder(trptr);

void nonrecpreorder(trptr);

void nonrecpostorder(trptr);

void nonrecinorder(trptr);

void main()

{

trptr t;

int x,ch;

clrscr();

t=NULL;

  printf("enter -1 to stop:");

while(1)

{

printf("\nEnter the element");

scanf("%d",&x);

if(x==-1)

break;

else

t=insertbst(t,x);

}

nonrecinorder(t);
getch();

}

trptr insertbst(trptr t,int x)

{

if(t==NULL)
```

```
{
t=(trptr)malloc(sizeof(struct treenode));
if(t==NULL)
printf("\nOut of space");
else
{
t->data=x;
t->lptr=t->rptr=NULL;
}
}
else if(x<t->data)
t->lptr=insertbst(t->lptr,x);
else
t->rptr=insertbst(t->rptr,x);
return(t);
}
void push(trptr s[30],trptr p)
{
if(tos>=29)
printf("\nOverflow on push");
else
{
tos++;
s[tos]=p;
}
}
trptr pop(trptr s[])
{
trptr y;
```

```
if(tos<=-1)

{

printf("\nUnderflow on pop");

return(NULL);

}

else

{

y=s[tos--];

return(y);

}

}
```

```
void   nonrecinorder(trptr t)
{
trptr s[30],p;
tos=-1;
if(t==NULL)
{
printf("\nTree is empty");
return;
}
else
{

push(s,t);

p=t->lptr;

while(tos>=-1)

{

while(p!=NULL)

{

push(s,p);

p=p->lptr;

}

if(tos!=-1)
```

```
{

p=pop(s);

printf("\t%d",p->data);

p=p->rptr;

}

else

break;

}

}

}
```

Result:

Non- Recursive traversals were performed over BST.

EXERCISE NO : 13

Write a program to implement two stacks using single array

Aim: A program to implement two stacks using single array.

Source Code:

```
#include<stdio.h>
Void main()
{
Int ch;
Int n,a[],x;
Printf(“enter ur choice 1-stack1 & 2- stack2”);
Scanf(“%d”,&ch);
Printf(“Enter ur choce push-1,pop-2,exit-3”);
Scanf(“%d”,&n);
Switch(n)
{
Case 1:
Printf(“enter element to be inserted”);
Scanf(“%d”,&x);
If(ch==1)
Push1(a,x);
Else
Push2(a,x);
Break;
Case 2:
If(ch==2)
X=Pop1();
Else
X=Pop2();
Break;
```

```
Case 3:
Exit(0);
}


}

void push1(int s[] ,int x)
{
if(tos2 –tos1= = 1)
printf(“stack is full :Overflowerror”);
else
s [++tos1] = x;
}
void push2(int s[],int x)
{
if( tos2-tos1 = = 1)
printf(“stack is full: Overflowerror”);
else
s[--tos2] = x;
}
int pop1[int s[])
{
if(tos1<=-1)
{
printf(“underflow on pop1”);
return(-1);
}
else
{
tos--;
return(s[tos1++]);
}
}
int pop2(int s[])
{
if(tos2>=n)
{
printf(“underlow on pop2”);
return(-1);
}
else
{
tos2++;
return(s[tos2-1]);
}
}
```

Result: Organizing 2 stacks using a single array was examined.

EXERCISE NO : 14

Write a program to Multiply 2 polynomials.

Aim: A program to multiply two polynomials.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
{
            float coef;
            int xp;
            struct node *next;
};
typedef struct node *tptr;
          void create(node *head)
          {
            printf("enter data(Enter 0 to stop):");
            scanf("%d",&head->xp);
            scanf("%d",&head->coef);
            if(head->xp==0)
            {
               head->next=NULL;
              head->coeff=0;
               return;
            }
            else
            {
                head->next=(node *)malloc(sizeof(struct node));
                create(head->next);
             }
          }

         void display(node *head)
         {
           if(head->next==NULL)
           {

              return;
            }
           else
             {  if(head->data==0)
                  return;
               printf("\n\t%d",head->xp);
               printf("\->%d",head->coef);
               display(head->next);
             }
         }

void main()
{
tptr p1,p2,p3;
       p1=(node *)malloc(sizeof(struct node));
       p2= (node *)malloc(sizeof(struct node));
       p3=polyprod(p1,p2);
       display(p1);
       display(p2);
```

```
    display(p3);
    }

tptr polyprod(tptr p1, tptr p2)
{
tptr t1,t2,p3;
int c,e;
p3 = (node*) malloc(sizeof(struct node));
p3->next=NULL;
for( t1 = p1->next ;t1! = NULL;t1 = t1->next)
for(t2 = p2->next;t2!= NULL;t2 = t2->next)
{
c = t1->coef * t2 ->coef;
e = t1->xp + t2->xp;
terminsert(p3,c,e);
}
return(p3);
}

    Void terminsert(tptr p3,int c, int e)
    {
     tptr temp;
        temp=p3;
    if(temp->next==NULL)
                    {
    temp->xp=e;
    temp->coef=c;
    }
    else
        {while(temp->next!=NULL)
                                {
                                 temp=temp->next;
                                }
                                 Newnode=(node*) malloc(sizeof(struct node));
                                 temp->next=newnode;
                                 newnode->next=NULL;
    temp->xp=e;
    temp->coef=c;

                        }
       }
```

Result: two polynomials were modified using linked lists.