1) SLIP1A

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch;
```

```
node *root,*temp,*ptr;
root=NULL;
do{
   temp=getnode();
   if(root==NULL)
          root=temp;
   else
   {
          ptr=root;
         while(ptr!=NULL)
         {
          if(temp->data<ptr->data)
          {
           if(ptr->lchild==NULL)
            {
             ptr->lchild=temp;
             break;
            }
            else
              ptr=ptr->lchild;
          }
          else
          {
            if(ptr->rchild==NULL)
            {
```

```
                    ptr->rchild=temp;
                    break;
                }
                else
                  ptr=ptr->rchild;
            }
     }//while
   }
     printf("\nDo you want to add more node(Y/N): ");
   //  scanf(" %c",&ch);
     ch=getche();
    }while(ch=='Y'||ch=='y');
    return root;
}
node *insert(node *root,node *temp)
{
  int ch;
  node *ptr;
   if(root==NULL)
            root=temp;
     else
     {
            ptr=root;
           while(ptr!=NULL)
           {
```

```
            if(temp->data<ptr->data)
            {
             if(ptr->lchild==NULL)
              {
               ptr->lchild=temp;
               break;
              }
             else
               ptr=ptr->lchild;
            }
            else
            {
               if(ptr->rchild==NULL)
               {
                       ptr->rchild=temp;
                       break;
               }
               else
                 ptr=ptr->rchild;
            }
      }//while
   }

    return root;
}
```

```
void display(node *ptr)
{
  if(ptr!=NULL)
  {
    display(ptr->lchild);
    printf("   %d",ptr->data);
    display(ptr->rchild);
  }
}
void main()
{
  int ch;
  node *root=NULL,*temp;
  clrscr();
  while(1)
  {
     printf("\n1:Create a BST.");
     printf("\n2:Insert element into a BST.");
     printf("\n3:Display");
     printf("\n4:Exit");
     printf("\nEnter the Choice:");
     scanf("%d",&ch);
     switch(ch)
     {
```

```
            case 1: root=create();
                    break;
            case 2: temp=getnode();
                    root=insert(root,temp);
                    break;
            case 3: display(root);
                    break;
            case 4: exit(0);
    }
 }
    getch();
}
```

2) SLIP2A

```
#include<stdio.h>
#include<conio.h>
#define MAX 50
struct STACK
{
     char stk[MAX];
     int top;
};
typedef struct STACK stack;
void initstack(stack *s)
{
```

```
  s->top=-1;
}
int isempty(stack *s)
{
  if(s->top==-1)
    return 1;
  else
    return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
          return 1;
  else
          return 0;
}
void push(stack *s,char data)
{
   s->top++;
   s->stk[s->top]=data;
}
char pop(stack *s)
{
   return(s->stk[s->top--]);
}
```

```
void main()
{
  stack s;
  char string[MAX];
  int i;
  clrscr();
  initstack(&s);
  printf("\nEnter the String: ");
  gets(string);
  for(i=0;string[i]!='\0';i++)
  {
     if(isfull(&s))
            printf("\nStack is FULL.");
     else
            push(&s,string[i]);
  }
 printf("\nReverse String:");
   while(!isempty(&s))
     printf("%c",pop(&s));
  getch();
}
```

3)SLIP 3A

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
   int data;
   struct NODE *next;
};
typedef struct NODE node;
```

```
int num;


node * getnode() //create a
node
{
   node *temp;
   temp=(node
*)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->next=NULL;
```

```
   return temp;
 }



node * create(node *list)
{
      node *temp,*last;
      int n,i;
      printf("\nEnter total nodes:
");
      scanf("%d",&n);
```

```
 for(i=0;i<n;i++)
 {
temp=getnode();
if(list==NULL)
    list=temp;
else
{
  for(last=list;last-
>next!=NULL;last=last->next);
   last->next=temp;
```

```
    }
     }
    return list;
}
void display(node *list)
{
   node *ptr;
   for(ptr=list;ptr!=NULL;ptr=ptr->next)
     printf("%d->",ptr->data);
   printf("NULL");
```

```
}
node * getnodenum(int digit)
//create a node
{
   node *temp;
   temp=(node
*)malloc(sizeof(node));
   temp->data=digit;
   temp->next=NULL;
   return temp;
 }
```

```
node * search(int val,node
*list1)
{
   node *ptr;
   int pos;
    for(ptr=list1;ptr!=NULL &&
ptr->data!=val;ptr=ptr->next);
    return ptr;
}
```

```
node * unionlist(node
*list1,node *list2)
{
   node
*lunion=NULL,*ptr1,*last,*tem
p,*ptr2,*ptr;

for(ptr1=list1;ptr1!=NULL;ptr1=
ptr1->next)
   {
```

```
    temp=getnodenum(ptr1->data);
    if(lunion==NULL)
      lunion=temp;
    else
   {
     for(last=lunion;last->next!=NULL;last=last->next);
      last->next=temp;
    }
  }
```

```
for(ptr2=list2;ptr2!=NULL;ptr2=
ptr2->next)
  {
      ptr=search(ptr2-
>data,list1);
      if(ptr==NULL)
      {
      temp=getnodenum(ptr2-
>data);
```

```
        for(last=lunion;last-
>next!=NULL;last=last->next);
          last->next=temp;
        }
   }
   return(lunion);
}
void main()
{
    int n,i;
```

```
    node
*list1=NULL,*list2=NULL,*last,*
lunion;
    clrscr();
    printf("\nCreate list1: ");
    list1=create(list1);
    printf("\nCreate List2: ");
    list2=create(list2);
    printf("\nList1 is: ");
    display(list1);
    printf("\nList2 is : ");
```

```
    display(list2);
    lunion=unionlist(list1,list2);
    printf("\nList after Union: ");
    display(lunion);
    getch();
  }
```

4) SLIP 4A

```
#include<stdio.h>
#include<conio.h>
struct NODE
```

```
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node
*)malloc(sizeof(node));
```

```
    printf("\nEnter the data: ");
    scanf("%d",&temp->data);
    temp->lchild=NULL;
    temp->rchild=NULL;
    return temp;
}

node *create()
{
  int ch;
```

```
node *root,*temp,*ptr;
root=NULL;
do{
   temp=getnode();
   if(root==NULL)
  root=temp;
   else
   {
  ptr=root;
 while(ptr!=NULL)
```

```
{
 if(temp->data<ptr->data)
 {
  if(ptr->lchild==NULL)
   {
    ptr->lchild=temp;
    break;
   }
   else
     ptr=ptr->lchild;
```

```
}
else
{
   if(ptr->rchild==NULL)
   {
  ptr->rchild=temp;
  break;
   }
   else
     ptr=ptr->rchild;
```

```
      }
      }//while
    }
      printf("\nDo you want to
add more node(Y/N): ");
    //  scanf(" %c",&ch);
      ch=getche();
     }while(ch=='Y'||ch=='y');
     return root;
}
void inorder(node *ptr)
```

```
{
  if(ptr!=NULL)
  {
    inorder(ptr->lchild);
    printf("   %d",ptr->data);
    inorder(ptr->rchild);
  }
}
void postorder(node *ptr)
{
```

```
  if(ptr!=NULL)
  {
    postorder(ptr->lchild);
    postorder(ptr->rchild);
    printf("   %d",ptr->data);
  }
}

void main()
{
```

```
  int ch;
  node *root=NULL;
  clrscr();
  while(1)
  {
     printf("\n1:Create a BST.");
     printf("\n2:Inorder of BST.");
     printf("\n3:Postorder of
BST");
     printf("\n4:Exit");
     printf("\nEnter the Choice:");
```

```
scanf("%d",&ch);
switch(ch)
{
case 1: root=create();
    break;
case 2: inorder(root);
    break;
case 3: postorder(root);
    break;
case 4: exit(0);
```

```
        }
    }
        getch();
}
```

5) SLIP 5A

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
```

```
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch;
  node *root,*temp,*ptr;
  root=NULL;
  do{
     temp=getnode();
     if(root==NULL)
             root=temp;
     else
     {
             ptr=root;
            while(ptr!=NULL)
            {
             if(temp->data<ptr->data)
             {
              if(ptr->lchild==NULL)
               {
```

```
                ptr->lchild=temp;
                break;
               }
               else
                 ptr=ptr->lchild;
              }
              else
              {
                if(ptr->rchild==NULL)
                {
                        ptr->rchild=temp;
                        break;
                }
                else
                  ptr=ptr->rchild;
             }
     }//while
   }
     printf("\nDo you want to add more node(Y/N): ");
   //  scanf(" %c",&ch);
     ch=getche();
    }while(ch=='Y'||ch=='y');
    return root;
}
void inorder(node *ptr)
```

```
{
 if(ptr!=NULL)
 {
  inorder(ptr->lchild);
  printf("   %d",ptr->data);
  inorder(ptr->rchild);
 }
}
void preorder(node *ptr)
{
 if(ptr!=NULL)
 {
  printf("   %d",ptr->data);
  preorder(ptr->lchild);
  preorder(ptr->rchild);
 }
}

void main()
{
 int ch;
 node *root=NULL;
 clrscr();
 while(1)
 {
```

```
    printf("\n1:Create a BST.");
    printf("\n2:Inorder of BST.");
    printf("\n3:Preorder of BST");
    printf("\n4:Exit");
    printf("\nEnter the Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
            case 1: root=create();
                    break;
            case 2: inorder(root);
                    break;
            case 3: preorder(root);
                    break;
            case 4: exit(0);
    }
}
    getch();
          }
```

6) SLIP5B

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

```
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnodenum(int digit) //create a
node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  temp->data=digit;
```

```
   temp->next=NULL;
   return temp;
 }
void createnumll(int num)
{
   int rem;
   node *temp;
  while(num>0)
  {
     rem=num%10;
     num=num/10;
//    printf("\nrem=%d
num=%d",rem,num);
```

```
    temp=getnodenum(rem);
     if(list==NULL)
        list=temp;
      else
      {
       temp->next=list;
       list=temp;
      }


  }
}
void display()
{
```

```
   node *ptr;
   for(ptr=list;ptr!=NULL;ptr=ptr->next)
     printf("%d->",ptr->data);
   printf("NULL");
}

void main()
{
    int num;
         clrscr();
    printf("\nEnter the No: ");
    scanf("%d",&num);
    createnumll(num);
```

```
    printf("\nLinked after separating the
digits of num: ");
    display();
    getch();
}
```

7) SLIP6A

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
```

```
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}
```

```
node *create()
{
  int ch;
  node *root,*temp,*ptr;
  root=NULL;
  do{
      temp=getnode();
      if(root==NULL)
       root=temp;
      else
      {
       ptr=root;
      while(ptr!=NULL)
```

```
{
 if(temp->data<ptr->data)
 {
  if(ptr->lchild==NULL)
   {
    ptr->lchild=temp;
    break;
   }
   else
     ptr=ptr->lchild;
  }
  else
  {
```

```
        if(ptr->rchild==NULL)
        {
         ptr->rchild=temp;
         break;
        }
        else
          ptr=ptr->rchild;
      }
     }//while
   }
     printf("\nDo you want to add more node(Y/N): ");
   //  scanf(" %c",&ch);
```

```
     ch=getche();
    }while(ch=='Y'||ch=='y');
    return root;
}
void postorder(node *ptr)
{
  if(ptr!=NULL)
  {
    postorder(ptr->lchild);
    postorder(ptr->rchild);
    printf("   %d",ptr->data);
  }
}
```

```
void preorder(node *ptr)
{
  if(ptr!=NULL)
  {
    printf("   %d",ptr->data);
    preorder(ptr->lchild);
    preorder(ptr->rchild);
  }
}

void main()
{
  int ch;
```

```
node *root=NULL;
clrscr();
while(1)
{
   printf("\n1:Create a BST.");
   printf("\n2:preorder of BST.");
   printf("\n3:Postorder of BST");
   printf("\n4:Exit");
   printf("\nEnter the Choice:");
   scanf("%d",&ch);
   switch(ch)
   {
    case 1: root=create();
```

```
            break;
        case 2: preorder(root);
            break;
        case 3: postorder(root);
            break;
        case 4: exit(0);
      }
   }
     getch();
}
```

8) SLIP7B

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

```
struct NODE
{
   int data;
   struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
```

```
   temp->next=NULL;
   return temp;
 }



void create()
{
   int n,i;
   node *temp;
   printf("\nEnter total nodes to be
creted");
   scanf("%d",&n);
```

```
   for(i=0;i<n;i++)
   {
      temp=getnode();
      if(list==NULL)
           list=temp;
      else
         last->next=temp;
      last=temp;
   }
}
void display()
{
   node *ptr;
```

```
   for(ptr=list;ptr!=NULL;ptr=ptr->next)
     printf("\t%d",ptr->data);
}

void count()
{
   node *ptr;
   int cnt;
   for(ptr=list,cnt=0;ptr!=NULL;ptr=ptr-
>next,cnt++);
   printf("\ntotal no of Nodes=%d",cnt);
}
```

```
void main()
{
   clrscr();
   create();
   display();
   count();
   getch();
}
```

9)SLIP8A

```
#include<stdio.h>
#include<conio.h>
struct NODE
```

```
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
```

```
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch;
  node *root,*temp,*ptr;
  root=NULL;
  do{
      temp=getnode();
      if(root==NULL)
       root=temp;
```

```
else
{
 ptr=root;
while(ptr!=NULL)
{
 if(temp->data<ptr->data)
 {
  if(ptr->lchild==NULL)
   {
    ptr->lchild=temp;
    break;
   }
   else
```

```
    ptr=ptr->lchild;
  }
  else
  {
    if(ptr->rchild==NULL)
    {
     ptr->rchild=temp;
     break;
    }
    else
      ptr=ptr->rchild;
  }
}//while
```

```
    }
      printf("\nDo you want to add more
node(Y/N): ");
    //  scanf(" %c",&ch);
      ch=getche();
     }while(ch=='Y'||ch=='y');
     return root;
}

void display(node *ptr)
{
  if(ptr!=NULL)
  {
```

```
    display(ptr->lchild);
    printf("   %d",ptr->data);
    display(ptr->rchild);
  }
}
int search(node *ptr,int val)
{
     while(ptr!=NULL)
     {
      if(ptr->data==val)
         return 1;
      if(val<ptr->data)
          ptr=ptr->lchild;
```

```
      else
          ptr=ptr->rchild;
      }
      return 0;
}

void main()
{
  int ch,val,resp;
  node *root=NULL,*temp    ;
  clrscr();
  while(1)
  {
```

```
    printf("\n1:Create a BST.");
    printf("\n2:Display");
    printf("\n3:search");
    printf("\n4:Exit");
    printf("\nEnter the Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
     case 1: root=create();
          break;
     case 3:printf("\nEnter the value to search:");
          scanf("%d",&val);
```

```
            resp=search(root,val);
            if(resp==1)
               printf("\nValue is found in
Tree.");
            else
               printf("\nValue not found in
Tree.");
            break;
        case 2: display(root);
             break;
        case 4: exit(0);
      }
   }
```

```
    getch();
}
```

10) SLIP9A

```
#include<stdio.h>
#include<conio.h>
# define MAX 30
struct STACK
{
   double stk[MAX];
   int top;
};
typedef struct STACK stack;
```

```
//initialize the stack
void initstack(stack *s)
{
   int i;
   for(i=0;i<MAX;i++)
    s->stk[i]=0;
   s->top=-1;
}
int isempty(stack *s)
{
    if(s->top==-1)
      return 1;
    else
```

```
      return 0;
}
int isfull(stack *s)
{
   if(s->top==MAX-1)
     return 1;
   else
     return 0;
}
      void push(stack *s,double data)
{
//   printf("\ndata=%f",data);
   s->top++;
```

```
   s->stk[s->top]=data;


}
double pop(stack *s)
{
   double val;
   val=s->stk[s->top];
   s->top--;
   return(val);  //return(s->stk[s->top--]);
}
void display(stack *s)
{
   int i;
```

```
   printf("\nStack: ");
   for(i=0;i<=s->top;i++)
      printf("\t%f",s->stk[i]);
}
int isdigit(char symb)
{
    if(symb>='0' && symb<='9')
      return 1;
    else
      return 0;
}
double eval(double opnd1,double
opnd2,char symb)
```

```
{
  switch(symb)
  {
    case '+': return(opnd1+opnd2);
    case '-': return(opnd1-opnd2);
    case '*': return(opnd1*opnd2);
    case '/': return(opnd1/opnd2);
    case '^': return(pow(opnd1,opnd2));
  }

}

int isoperator(char symbol)
```

```
{
    if(symbol=='+' || symbol=='-' ||
symbol=='*'|| symbol=='/'||
symbol=='^')
        return 1;
    else
        return 0;
}
char* convert(char *expr)
{
 int i,j,scnt=0,symval[10],flag=0;
  char symbol[10];                //01234567
scnt=0,1,2,3
                    //expr=42+54-*\0
```

```
symbol[scnt++]=expr[0];
//symbol[0]=a  [1]=b  [2]=c
printf("\nEnter the Value of %c:
",expr[0]);
scanf("%d",&symval[0]);
//symval[0]=4  [1]=2  [2]=5
expr[0]=symval[0]+48;
for(i=1;expr[i]!='\0';i++)      //i=1,2,3,4
flag=1
  {
   flag=0;
   if(isoperator(expr[i])==0)
    {
      for(j=0;j<scnt;j++)       //j=0
```

```
        {
         if(expr[i]==symbol[j])
         {
             flag=1;
             expr[i]=symval[j]+48;
             break;
         }
        }
        if(flag!=1)
        {
             symbol[scnt]=expr[i];
             printf("\nEnter the Value of %c:
",expr[i]);
```

```
            scanf("%d",&symval[scnt]);
            expr[i]=symval[scnt]+48;
            scnt++;
        }


      }
    }
    printf("\n Expression after converting into digit form=%s",expr);
    return expr;
}
double posteval(char *expr)
{
```

```
stack s;
int i,j,scnt=0,symval[10],flag=0;
char symbol[10];
double opnd1,opnd2,result;
initstack(&s);
expr=convert(expr);
for(i=0;expr[i]!='\0';i++)
{

    if(isdigit(expr[i])==1)
     push(&s,expr[i]-48);
    else
    {
```

```
      opnd2=pop(&s);
      opnd1=pop(&s);
      result=eval(opnd1,opnd2,expr[i]);
      push(&s,result);
      }
   }
   return(pop(&s));
}
void main()
{
    char expr[MAX];
    clrscr();
    printf("\nEnter the Expression: ");
```

```
    scanf(" %s",expr);
    printf("\nResult after postfix
Evaluation=%f",posteval(expr));
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
   int data;
   struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
```

```
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->next=NULL;
   return temp;
 }



void create()
{
   int n,i;
   node *temp;
```

```
   printf("\nEnter total nodes to be
creted");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      temp=getnode();
      if(list==NULL)
            list=temp;
      else
         last->next=temp;
      last=temp;
   }
}
```

```
void display(node *list)
{
   node *ptr;
   for(ptr=list;ptr!=NULL;ptr=ptr->next)
     printf("%d->",ptr->data);
   printf("NULL");
}
node * reverse()
{
```

```
  node *ptr,*prev,*revlist=NULL;
 while(list!=NULL)
 {
  for(ptr=list,prev=list;ptr-
>next!=NULL;prev=ptr,ptr=ptr->next);
  if(ptr==list && prev==list)
    list=NULL;
  prev->next=NULL;
   if(revlist==NULL)
    revlist=ptr;
   else
    last->next=ptr;
   last=ptr;
```

```
  }
   return(revlist);
}
void main()
{
    int ch,val;
    node *revlist;
    clrscr();
    create();
    display(list);
    revlist=reverse();
    printf("\nList after reversing: ");
    display(revlist);
```

```
    printf("\noriginal list: ");
    display(list);
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
```

```
{
  int n,i;
  node *temp;
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}




void display(node *list)
{
  node *ptr;
```

```
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("NULL");
}
node * getnodenum(int num) //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  temp->data=num;
  temp->next=NULL;
  return temp;
 }
```

```
node * reverse()
{
  node *ptr,*temp,*revlist=NULL;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
```

```
  {
        temp=getnodenum(ptr->data);
        temp->next=revlist;
        revlist=temp;
  }
  return(revlist);
}
void main()
{
   int ch,val;
   node *revlist;
   clrscr();
   create();
   revlist=reverse();
   printf("\nList after reversing: ");
   display(revlist);
   printf("\noriginal list: ");
   display(list);
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
```

```
  node *temp;
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
          if(list==NULL)
                  list=temp;
          else
            last->next=temp;
          last=temp;
  }
}




void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}
void search(int val)
{
```

```
   node *ptr;
   int pos;
    for(ptr=list,pos=1;ptr!=NULL && ptr->data!=val;ptr=ptr->next,pos++);
    if(ptr->data==val)
       printf("Value is found in LL at %d position.",pos);
    if(ptr==NULL)
       printf("Value not found.");

}



void insertend()
{
    node *temp;
    temp=getnode();
    last->next=temp;
    last=temp;
}
void main()
{
    int ch,val;
    clrscr();

    while(1)
    {
```

```
            printf("\n1:Create the Linked list.");
            printf("\n2:Display the Linked List.");
            printf("\n3:Search .");
            printf("\n4: Insert at Last Position.");
            printf("\n5:Exit.");
            printf("\nEnter the choice: ");
            scanf("%d",&ch);
            switch(ch)
            {
              case 1: create();
                      break;
              case 2: display();
                      break;
              case 3: printf("\nEnter the values to be searched: ");
                      scanf("%d",&val);
                      search(val);
                      break;
              case 4:insertend();
                     break;
              case 5:exit();
            }
    }
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;

node *front=NULL,*rear=NULL;
int isempty()
{
  if(front==NULL && rear==NULL)
    return 1;
  else
     return 0;
}
node * getnodenum(int data)
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   temp->data=data;
   temp->next=NULL;
   return temp;
```

```
}
void add(int data)
{
  node *temp;
  temp=getnodenum(data);
  if(front==NULL)
   {
        rear=temp;
        front=temp;
   }
   else
    {
         rear->next=temp;
         rear=temp;
    }
}

int deleteq()
{
 node *ptr;
 ptr=front;
 val=front->data;
 if(front->next!=NULL)
   front=front->next;
 else
```

```
  {
     front=NULL;
     rear=NULL;
  }
  free(ptr);
  return(val);
}
void display()
{
  node *ptr;
  for(ptr=front;ptr!=rear;ptr=ptr->next)
     printf(" %d",ptr->data);
  printf(" %d",ptr->data);
}

void main()
{
   int ch,data;
   clrscr();
   while(1)
   {
     printf("\n1: Add.");
     printf("\n2: Delete");
     printf("\n3: Display.");
     printf("\n4: Exit");
```

```
    printf("\nEnter the Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1:printf("\nEnter the data to insert in queue:");
                scanf("%d",&data);
                add(data);
                break;
      case 2: if(isempty())
                      printf("\nQueue is Empty!");
                else
                       printf("\nDeleted element is: %d",deleteq());
                break;
      case 3:
                      if(isempty())
                        printf("\nQueue is Empty!");
                else
                       display();
                break;
      case 4: exit(1);
    }
  }
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
# define MAX 50
struct STACK
{
  char stk[MAX];
  int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
   s->top=-1;
}
int isempty(stack *s)
{
   if(s->top==-1)
     return 1;
   else
     return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
```

```
    return 1;
  else
    return 0;
}
void push(stack *s,char data)
{
  s->top++;
  s->stk[s->top]=data;

}
char pop(stack *s)
{
  char val;
  val=s->stk[s->top];
  s->top--;
  return(val);  //return(s->stk[s->top--]);
}
char gettop(stack *s)
{
 return(s->stk[s->top]);
}
void display(stack *s)
{
  int i;
//   printf("\nStack Content: ");
```

```
  for(i=0;i<=s->top;i++)
     printf("%c",s->stk[i]);
}

int isoperator(char symbol)
{
  if(symbol=='+' || symbol=='-' || symbol=='*'|| symbol=='/'|| symbol=='^')
                return 1;
            else
                return 0;
}
int priority(char oper)
            {
  if(oper=='^')
            return 4;
  else if(oper=='/' || oper=='*')
    return 3;
  else if(oper=='+'||oper=='-')
    return 2;
  else
    return 1;
}
int checktop(char topsymb)
{
  if(topsymb=='(' || topsymb=='{' || topsymb=='['
```

```
            return 1;
            else
            return 0;
}
void inpostfix(char *E)
{
  int i,j=0,flag;
  char symbol,postfix[MAX],topsymb;
  stack s;
  initstack(&s);
  printf("Symbol  postfix\tstack");
  for(i=0;E[i]!='\0';i++)
  {
     symbol=E[i];
     if(symbol=='(' || symbol=='{' || symbol=='['
                 push(&s,symbol);
    else if(isoperator(symbol)==1)
       {
              while(priority(gettop(&s))>=priority(symbol))
                 postfix[j++]=pop(&s);
              push(&s,symbol);

       }

    else if(symbol==')' || symbol=='}' || symbol==']')
```

```
                {
                                topsymb=pop(&s);
                                while(!checktop(topsymb))
                                {
                                        postfix[j++]=topsymb;
                                        topsymb=pop(&s);
                                }
                }

  else
        postfix[j++]=symbol;
  postfix[j]='\0';
  printf("\n%c   %s\t\t",symbol,postfix);
  display(&s);
 getch();

}
  while(!isempty(&s))
  {
    postfix[j++]=pop(&s);
    postfix[j]='\0';
    printf("\n%c %s\t\t",symbol,postfix);
    display(&s);
  }
```

```
 //display(&s);
  printf("\nPostfix Expression: %s",postfix);
}
main()
{
    char expr[MAX];
clrscr();
    printf("\nEnter the Infix Expresion: ");
    scanf(" %s",expr);
   inpostfix(expr);
getch();
}
```

```
#include<stdio.h>
#include<conio.h>
# define MAX 50
struct STACK
{
  char stk[MAX];
  int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
   s->top=-1;
}
int isempty(stack *s)
{
   if(s->top==-1)
     return 1;
   else
     return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
```

```
    return 1;
  else
    return 0;
}
void push(stack *s,char data)
{
  s->top++;
  s->stk[s->top]=data;

}
char pop(stack *s)
{
  char val;
  val=s->stk[s->top];
  s->top--;
  return(val);  //return(s->stk[s->top--]);
}
char gettop(stack *s)
{
 return(s->stk[s->top]);
}
void display(stack *s)
{
  int i;
//   printf("\nStack Content: ");
```

```
  for(i=0;i<=s->top;i++)
     printf("%c",s->stk[i]);
}

int isoperator(char symbol)
{
  if(symbol=='+' || symbol=='-' || symbol=='*'|| symbol=='/'|| symbol=='^')
                return 1;
            else
                return 0;
}
int priority(char oper)
            {
  if(oper=='^')
            return 4;
  else if(oper=='/' || oper=='*')
    return 3;
  else if(oper=='+'||oper=='-')
    return 2;
  else
    return 1;
}
int checktop(char topsymb)
{
  if(topsymb=='(' || topsymb=='{' || topsymb=='['
```

```
            return 1;
            else
            return 0;
}
void inpostfix(char *E)
{
  int i,j=0,flag;
  char symbol,postfix[MAX],topsymb;
  stack s;
  initstack(&s);
  printf("Symbol  postfix\tstack");
  for(i=0;E[i]!='\0';i++)
  {
     symbol=E[i];
     if(symbol=='(' || symbol=='{' || symbol=='['
                 push(&s,symbol);
    else if(isoperator(symbol)==1)
       {
              while(priority(gettop(&s))>=priority(symbol))
                 postfix[j++]=pop(&s);
              push(&s,symbol);

       }

    else if(symbol==')' || symbol=='}' || symbol==']')
```

```
                {
                                topsymb=pop(&s);
                                while(!checktop(topsymb))
                                {
                                        postfix[j++]=topsymb;
                                        topsymb=pop(&s);
                                }
                }

  else
        postfix[j++]=symbol;
  postfix[j]='\0';
  printf("\n%c   %s\t\t",symbol,postfix);
  display(&s);
 getch();

}
  while(!isempty(&s))
  {
    postfix[j++]=pop(&s);
    postfix[j]='\0';
    printf("\n%c %s\t\t",symbol,postfix);
    display(&s);
  }
```

```
 //display(&s);
  printf("\nPostfix Expression: %s",postfix);
}
main()
{
    char expr[MAX];
clrscr();
    printf("\nEnter the Infix Expresion: ");
    scanf(" %s",expr);
   inpostfix(expr);
getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
   struct NODE *prev;
   int data;
   struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node *getnode()
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->prev=NULL;
  temp->next=NULL;
  return temp;
}
void create()
{
    int n,i;
    node *temp;
    printf("\nEnter total nodes: ");
```

```
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
      temp=getnode();
      if(list==NULL)
            list=temp;
      else
       {
             last->next=temp;
             temp->prev=last;
       }
       last=temp;
    }
}

void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
    printf("->NULL");
}
void displayodd()
{
   node *ptr;
```

```
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
  {
    if((ptr->data%2)!=0)
          printf("%d->",ptr->data);
  }
  printf("->NULL");
}

void main()
{
  clrscr();
  create();
  printf("\nDoubly LL: ");
  display();
  printf("\nOdd value in Doubly LL: ");
  displayodd();
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  char data;
  struct NODE *next;
};
typedef struct NODE node;
node *top=NULL;
node *getnodenum(char data)
{
  node *temp;
  temp=(node*)malloc(sizeof(node));
  temp->data=data;
  temp->next=NULL;
  return(temp);
}
int isempty()
{
  if(top==NULL)
    return 1;
  else
    return 0;
}
void push(char data)
```

```
{
   node *temp;
   temp=getnodenum(data);
   temp->next=top;
   top=temp;
}
char pop()
{
  char val;
  node *ptr;
  ptr=top;
  val=ptr->data;
  top=ptr->next;
  free(ptr);
  return val;
}

void main()
{
  int i;
  char string[50];
  clrscr();
 printf("\nEnter the String: ");
  gets(string);
  for(i=0;string[i]!='\0';i++)
```

```
            push(string[i]);


 printf("\nReverse String:");
   while(!isempty())
      printf("%c",pop());
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *top=NULL;
node *getnodenum(int data)
{
  node *temp;
  temp=(node*)malloc(sizeof(node));
  temp->data=data;
  temp->next=NULL;
  return(temp);
}
int isempty()
{
  if(top==NULL)
    return 1;
  else
    return 0;
}
void push(int data)
{
  node *temp;
  temp=getnodenum(data);
  temp->next=top;
  top=temp;
}
int pop()
{
 int val;
 node *ptr;
 ptr=top;
 val=ptr->data;
 top=ptr->next;
 free(ptr);
 return val;
}
void display()
{
  node *ptr;
  for(ptr=top;ptr!=NULL;ptr=ptr->next)
     printf("\t%d",ptr->data);
}
```

```
void main()
{
 int ch,data;
 clrscr();
 while(1)
 {
 printf("\n1:PUSH.");
 printf("\n2:POP.");
 printf("\n3:Display.");
 printf("\n4:Exit.");
 printf("\nEnter the Choice:");
 scanf("%d",&ch);
 switch(ch)
 {
   case 1: printf("\nEnter the data:");
           scanf("%d",&data);
           push(data);
           break;

   case 2:  if(isempty())
                printf("\nstack is empty.");
            else
              printf("\nPopped data from stack: %d",pop());
            break;
   case 3:   if(isempty())
                        printf("\nStack is Empty.");
            else
                        display();
           break;
   case 4:exit(0);
 }
 }
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
#define MAX 50
struct STACK
{
      char stk[MAX];
      int top;
};
typedef struct STACK stack;
void initstack(stack *s)
{
  s->top=-1;
}
int isempty(stack *s)
{
  if(s->top==-1)
    return 1;
  else
    return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
            return 1;
  else
```

```
            return 0;
}
void push(stack *s,char data)
{
   s->top++;
   s->stk[s->top]=data;
}
char pop(stack *s)
{
   return(s->stk[s->top--]);
}
void main()
{
  stack s;
  char string[MAX];
  int i;
  clrscr();
  initstack(&s);
  printf("\nEnter the String: ");
  gets(string);
  strcat(string," ");
  printf("\nReverse String:");
  for(i=0;string[i]!='\0';i++)
  {
    if(string[i]!=' ')
```

```
   {
    if(isfull(&s))
            printf("\nStack is FULL.");
    else
            push(&s,string[i]);
   }
   else
   {
    while(!isempty(&s))
             printf("%c",pop(&s));
    printf(" ");
   }
 }
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}
void search(int val)
{
  node *ptr;
  int pos;
   for(ptr=list,pos=1;ptr!=NULL && ptr->data!=val;ptr=ptr->next,pos++);
   if(ptr->data==val)
      printf("Value is found in LL at %d position.",pos);
```

```
    if(ptr==NULL)
       printf("Value not found.");

}

void main()
{
    int ch,val;
    clrscr();
    create();
    display();
    printf("\nEnter the values to be searched: ");
    scanf("%d",&val);
    search(val);
    getch();
}
```

```
        #include<stdio.h>
#include<conio.h>
# define MAX 30
struct STACK
{
  double stk[MAX];
  int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
  int i;
  for(i=0;i<MAX;i++)
   s->stk[i]=0;
  s->top=-1;
}
int isempty(stack *s)
{
   if(s->top==-1)
     return 1;
   else
     return 0;
}
```

```
int isfull(stack *s)
{
  if(s->top==MAX-1)
    return 1;
  else
    return 0;
}
void push(stack *s,double data)
{
//   printf("\ndata=%f",data);
  s->top++;
  s->stk[s->top]=data;

}
double pop(stack *s)
{
  double val;
  val=s->stk[s->top];
  s->top--;
  return(val);  //return(s->stk[s->top--]);
}
void display(stack *s)
{
  int i;
  printf("\nStack: ");
```

```
   for(i=0;i<=s->top;i++)
      printf("\t%f",s->stk[i]);
}
int isdigit(char symb)
{
    if(symb>='0' && symb<='9')
      return 1;
    else
      return 0;
}
double eval(double opnd1,double opnd2,char symb)
{
  switch(symb)
  {
    case '+': return(opnd1+opnd2);
    case '-': return(opnd1-opnd2);
    case '*': return(opnd1*opnd2);
    case '/': return(opnd1/opnd2);
    case '^': return(pow(opnd1,opnd2));
  }

}
double posteval(char *expr)
{
  stack s;
```

```
 int i;
 double opnd1,opnd2,result;
 initstack(&s);
 for(i=0;expr[i]!='\0';i++)
 {

    if(isdigit(expr[i])==1)
            push(&s,expr[i]-48);
    else
    {
           opnd2=pop(&s);
           opnd1=pop(&s);
           result=eval(opnd1,opnd2,expr[i]);
           push(&s,result);
    }
 }
 display(&s);
 return(pop(&s));
}
void main()
{
   char expr[MAX];
   clrscr();
   printf("\nEnter the Expression: ");
   scanf(" %s",expr);
```

```
    printf("Result after postfix Evaluation=%f",posteval(expr));
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
int cntleaf,totalnodes;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch;
  node *root,*temp,*ptr;
```

```
root=NULL;
do{
   temp=getnode();
   if(root==NULL)
          root=temp;
   else
   {
          ptr=root;
         while(ptr!=NULL)
         {
          if(temp->data<ptr->data)
          {
           if(ptr->lchild==NULL)
            {
             ptr->lchild=temp;
             break;
            }
            else
              ptr=ptr->lchild;
           }
           else
           {
             if(ptr->rchild==NULL)
             {
                    ptr->rchild=temp;
```

```
                    break;
                }
                else
                  ptr=ptr->rchild;
              }
      }//while
    }
      printf("\nDo you want to add more node(Y/N): ");
    //  scanf(" %c",&ch);
      ch=getche();
     }while(ch=='Y'||ch=='y');
     return root;
}
void display(node *ptr)
{
  if(ptr!=NULL)
  {
    display(ptr->lchild);
    printf("   %d",ptr->data);
    display(ptr->rchild);
  }
}
void leafcount(node *ptr)
{
   if(ptr!=NULL)
```

```
  {
     leafcount(ptr->lchild);
     if(ptr->lchild==NULL && ptr->rchild==NULL)
            cntleaf++;
     leafcount(ptr->rchild);
  }
}
void totalcount(node *ptr)
{
  if(ptr!=NULL)
  {
     totalnodes++;
     totalcount(ptr->lchild);

     totalcount(ptr->rchild);
  }
}

void main()
{
 int ch;
 node *root=NULL;
 clrscr();
 while(1)
 {
```

```
  printf("\n1:Create a BST.");
  printf("\n2:Display BST.");
  printf("\n3:Number of Nodes");
  printf("\n4:Degree of Tree");
  printf("\n5:Leaf Node Count");
  printf("\n6:Exit");
  printf("\nEnter the Choice:");
  scanf("%d",&ch);
  switch(ch)
  {
        case 1: root=create();
                 break;
        case 2: display(root);
                 break;
        case 3:totalcount(root);
                printf("\nTotal node: %d",        totalnodes);
                                  break;
        case 4: break;
        case 5:leafcount(root);
                printf("\nTotal Leaf node: %d",cntleaf);
                break;
        case 6:exit(0);
  }
}
  getch();
```

}

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("NULL");
}
/*void displayalternate()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
  {
    printf("%d->",ptr->data);
```

```
      ptr=ptr->next;
   }
   printf("NULL");
} */
void displayalternate()
{
   node *ptr;
   int cnt;
   for(ptr=list,cnt=1;ptr!=NULL;ptr=ptr->next,cnt++)
   {
    if(cnt%2!=0)
      printf("%d->",ptr->data);
   }
   printf("NULL");
}
void main()
{
    int ch,val;
    clrscr();
    create();
    display();
    printf("\nAlternate nodes in LL: ");
    displayalternate();
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
#define MAX 50
struct STACK
{
      char stk[MAX];
      int top;
};
typedef struct STACK stack;
void initstack(stack *s)
{
  s->top=-1;
}
int isempty(stack *s)
{
  if(s->top==-1)
    return 1;
  else
    return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
            return 1;
  else
```

```
            return 0;
}
void push(stack *s,char data)
{
   s->top++;
   s->stk[s->top]=data;
}
char pop(stack *s)
{
   return(s->stk[s->top--]);
}
void main()
{
  stack s;
  char string[50],revstr[50];
  int i,j=0;
  clrscr();
  initstack(&s);
  printf("\nEnter the String: ");
  gets(string);
  for(i=0;string[i]!='\0';i++)
  {
     if(isfull(&s))
            printf("\nStack is FULL.");
     else
```

```
            push(&s,string[i]);
  }
     while(!isempty(&s))
             revstr[j++]=pop(&s);
     revstr[j]='\0';

     if(strcmp(string,revstr)==0)
              printf("\nString is Palindrom.");
     else
              printf("\nString is not Palindrom.");
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
          if(list==NULL)
                  list=temp;
          else
            last->next=temp;
          last=temp;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}

void swap()
{
  node *ptr,*ptr1;
  int pos,m,n,temp;
  printf("\nEnter the position of mth and nth to be swapped");
  scanf("%d%d",&m,&n);
```

```
  for(ptr=list,pos=1;ptr!=NULL && pos<m;ptr=ptr->next,pos++);
  for(ptr1=list,pos=1;ptr1!=NULL && pos<n;ptr1=ptr1->next,pos++);
  if(ptr!=NULL && ptr1!=NULL)
  {
    temp=ptr->data;
    ptr->data=ptr1->data;
    ptr1->data=temp;
  }
  else
    printf("\nPosition is invalid");
}



void main()
{
   clrscr();
   create();
   display();
   swap();
   printf("\nLinked list after Swapping:  ");
   display();
   getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  char data;
  struct NODE *next;
};
typedef struct NODE node;
node *top=NULL;
node *getnodenum(char data)
{
  node *temp;
  temp=(node*)malloc(sizeof(node));
  temp->data=data;
  temp->next=NULL;
  return(temp);
}
int isempty()
{
  if(top==NULL)
    return 1;
  else
    return 0;
}
void push(char data)
```

```
{
   node *temp;
   temp=getnodenum(data);
   temp->next=top;
   top=temp;
}
char pop()
{
  char val;
  node *ptr;
  ptr=top;
  val=ptr->data;
  top=ptr->next;
  free(ptr);
  return val;
}

void main()
{
  char string[50],revstr[50];
  int i,j=0;
  clrscr();
  printf("\nEnter the String: ");
  gets(string);
  for(i=0;string[i]!='\0';i++)
```

```
            push(string[i]);


  while(!isempty())
    revstr[j++]=pop();
  revstr[j]='\0';


     if(strcmp(string,revstr)==0)
            printf("\nString is Palindrom.");
     else
            printf("\nString is not Palindrom.");
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}

void count()
{
  int nonzero=0,even=0,odd=0;
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
  {
```

```
      if(ptr->data!=0)
            nonzero++;
      if(ptr->data%2==0)
            even++;
      else
            odd++;
   }
   printf("\nTotal non zero Numbers= %d",nonzero);
   printf("\nTotal even Numbers= %d",even);
   printf("\nTotal odd Numbers= %d",odd);
}
void main()
{
    int ch,val;
    clrscr();
    create();
    display();
    count();
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}




void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}
void search(int val)
{
  node *ptr;
```

```
   int pos;
    for(ptr=list,pos=1;ptr!=NULL && ptr->data!=val;ptr=ptr->next,pos++);
    if(ptr->data==val)
       printf("Value is found in LL at %d position.",pos);
    if(ptr==NULL)
       printf("Value not found.");

}



void insertend()
{
    node *temp;
    temp=getnode();
    last->next=temp;
    last=temp;
}
node * deletepos()
{
    int pos,i;
    node *ptr,*prev;
    printf("\nenter the Position for deleting node: ");
    scanf("%d",&pos);
    if(pos==1)
    {
```

```
      ptr=list;
      list=ptr->next;
      free(ptr);
    }
    else
    {
      for(ptr=list,prev=list,i=1;ptr!=NULL && i<pos;prev=ptr,ptr=ptr->next,i++);
      if(ptr!=NULL)
      {
              prev->next=ptr->next;
              free(ptr);
      }
      else
            printf("\nPosition not found: ");
    }
    return list;
}

void main()
{
    int ch,val;
    clrscr();

    while(1)
    {
```

```
        printf("\n1:Create the Linked list.");
        printf("\n2:Display the Linked List.");
        printf("\n3: Insert at Last Position.");
        printf("\n4: Delete by position");
        printf("\n5:Exit.");
        printf("\nEnter the choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
          case 1: create();
                  break;
          case 2: display();
                  break;
          case 4: list=deletepos();
                  break;
          case 3:insertend();
                 break;
          case 5:exit();
        }
    }
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  double data;
  struct NODE *next;
};
typedef struct NODE node;
node *top=NULL;
node *getnodenum(double data)
{
  node *temp;
  temp=(node*)malloc(sizeof(node));
  temp->data=data;
  temp->next=NULL;
  return(temp);
}



int isempty()
{
  if(top==NULL)
    return 1;
  else
```

```
    return 0;
}
void push(double data)
{
  node *temp;
  temp=getnodenum(data);
  temp->next=top;
  top=temp;
}
double pop()
{
 double val;
 node *ptr;
 ptr=top;
 val=ptr->data;
 top=ptr->next;
 free(ptr);
 return val;
}
int isdigit(char symb)
{
   if(symb>='0' && symb<='9')
     return 1;
   else
     return 0;
```

```
}
double eval(double opnd1,double opnd2,char symb)
{
  switch(symb)
  {
    case '+': return(opnd1+opnd2);
    case '-': return(opnd1-opnd2);
    case '*': return(opnd1*opnd2);
    case '/': return(opnd1/opnd2);
    case '^': return(pow(opnd1,opnd2));
  }

}
double posteval(char *expr)
{
  int i;
  double opnd1,opnd2,result;
  for(i=0;expr[i]!='\0';i++)
  {

     if(isdigit(expr[i])==1)
              push(expr[i]-48);
     else
     {
            opnd2=pop();
```

```
            opnd1=pop();
            result=eval(opnd1,opnd2,expr[i]);
            push(result);
      }
  }
  return(pop());
}
void main()
{
    char expr[50];
    clrscr();
    printf("\nEnter the Expression: ");
    scanf(" %s",expr);
    printf("Result after postfix Evaluation=%f",posteval(expr));
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
        if(list==NULL)
                list=temp;
        else
          last->next=temp;
        last=temp;
  }
}




void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("NULL");
}
node * removelastaddfirst()
{
```

```
   node *ptr,*prev;
   for(ptr=list,prev=list;ptr->next!=NULL;prev=ptr,ptr=ptr->next);
   prev->next=NULL;
   ptr->next=list;
   list=ptr;
   return(list);
}
void main()
{
    int ch,val;
    clrscr();
    create();
    display();
    list=removelastaddfirst();
    printf("\nList after removing last node and adding at first: ");
    display();
 getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;

node *list=NULL,*last;
node * getnode()
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
}

void create()
{
  int n,i;
  node *temp;
  printf("\nEnter the no. of nodes: ");
```

```
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
    if(list==NULL)
            list=temp;
    else
            last->next=temp;
    temp->next=list;
           last=temp;
  }
}
void display()
{
 node *ptr;
 for(ptr=list;ptr->next!=list;ptr=ptr->next)
   printf("%d->",ptr->data);
 printf("%d->",ptr->data);
  printf("NULL");
}
void main()
{
  clrscr();
  create();
  display();
```

```
    getch();
}
```

```
        #include<stdio.h>
#include<conio.h>
# define MAX 50
struct STACK
{
  char stk[MAX];
  int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
   s->top=-1;
}
int isempty(stack *s)
{
   if(s->top==-1)
     return 1;
   else
     return 0;
}
int isfull(stack *s)
{
  if(s->top==MAX-1)
```

```
    return 1;
  else
    return 0;
}
void push(stack *s,char data)
{
  s->top++;
  s->stk[s->top]=data;

}
char pop(stack *s)
{
  char val;
  val=s->stk[s->top];
  s->top--;
  return(val);  //return(s->stk[s->top--]);
}
char gettop(stack *s)
{
 return(s->stk[s->top]);
}
void display(stack *s)
{
  int i;
//   printf("\nStack Content: ");
```

```
  for(i=0;i<=s->top;i++)
     printf("%c",s->stk[i]);
}

int isoperator(char symbol)
{
  if(symbol=='+' || symbol=='-' || symbol=='*'|| symbol=='/'|| symbol=='^')
                return 1;
            else
                return 0;
}
int priority(char oper)
            {
  if(oper=='^')
            return 4;
  else if(oper=='/' || oper=='*')
    return 3;
  else if(oper=='+'||oper=='-')
    return 2;
  else
    return 1;
}
int checktop(char topsymb)
{
  if(topsymb==')' || topsymb=='}' || topsymb==']')
```

```
            return 1;
            else
            return 0;
}
void inprefix(char *E)
{
  int i,j=0,len;
  char symbol,prefix[MAX],topsymb;
  stack s;
  initstack(&s);
  len=strlen(E);
  printf("Symbol  prefix\tstack");
  for(i=len-1;i>=0;i--)
  {
     symbol=E[i];
     if(symbol==')' || symbol=='}' || symbol==']')
                 push(&s,symbol);
    else if(isoperator(symbol)==1)
       {
              while(priority(gettop(&s))>priority(symbol))
                 prefix[j++]=pop(&s);
              push(&s,symbol);

       }
```

```
 else if(symbol=='(' || symbol=='{' || symbol=='[')
                    {
                                        topsymb=pop(&s);
                                        while(!checktop(topsymb))
                                        {
                                                prefix[j++]=topsymb;
                                                topsymb=pop(&s);
                                        }
                    }

  else
         prefix[j++]=symbol;
  prefix[j]='\0';
  printf("\n%c   %s\t\t",symbol,prefix);
  display(&s);
 getch();

}
 while(!isempty(&s))
 {
   prefix[j++]=pop(&s);
   prefix[j]='\0';
   printf("\n%c %s\t\t",symbol,prefix);
   display(&s);
 }
```

```
   strrev(prefix);
 //display(&s);
  printf("\nPrefix Expression: %s",prefix);
}
main()
{
    char expr[MAX];
clrscr();
    printf("\nEnter the Infix Expresion: ");
    scanf(" %s",expr);
   inprefix(expr);
getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;

int num;

node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }

node * create(node *list)
{
    node *temp,*last;
```

```
    int n,i;
    printf("\nEnter total nodes: ");
    scanf("%d",&n);

    for(i=0;i<n;i++)
    {
        temp=getnode();
        if(list==NULL)
                list=temp;
        else
        {
          for(last=list;last->next!=NULL;last=last->next);
           last->next=temp;
        }
    }
        return list;
}
void display(node *list)
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("NULL");
}
void main()
```

```
{
   int n,i;
   node *list1=NULL,*list2=NULL,*last;
   clrscr();
   printf("\nCreate list1: ");
   list1=create(list1);
   printf("\nCreate List2: ");
   list2=create(list2);
   printf("\nList1 is: ");
   display(list1);
   printf("\nList2 is : ");
   display(list2);
   for(last=list1;last->next!=NULL;last=last->next);
     last->next=list2;
   printf("\nList after concatenation is: ");
   display(list1);
   getch();
 }
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch;
  node *root,*temp,*ptr;
  root=NULL;
```

```
do{
   temp=getnode();
   if(root==NULL)
          root=temp;
   else
   {
          ptr=root;
         while(ptr!=NULL)
         {
          if(temp->data<ptr->data)
          {
           if(ptr->lchild==NULL)
            {
             ptr->lchild=temp;
             break;
            }
            else
              ptr=ptr->lchild;
          }
          else
          {
            if(ptr->rchild==NULL)
            {
                    ptr->rchild=temp;
                    break;
```

```
                }
                else
                  ptr=ptr->rchild;
              }
      }//while
    }
      printf("\nDo you want to add more node(Y/N): ");
    //  scanf(" %c",&ch);
      ch=getche();
     }while(ch=='Y'||ch=='y');
     return root;
}

void display(node *ptr)
{
  if(ptr!=NULL)
  {
    display(ptr->lchild);
    display(ptr->rchild);
    printf("   %d",ptr->data);
  }
}
int search(node *ptr,int val)
{
     while(ptr!=NULL)
```

```
    {
            if(ptr->data==val)
              return 1;
            if(val<ptr->data)
               ptr=ptr->lchild;
            else
               ptr=ptr->rchild;
    }
    return 0;
}

void main()
{
  int ch,val,resp;
  node *root=NULL,*temp          ;
  clrscr();
  while(1)
  {
    printf("\n1:Create a BST.");
    printf("\n2:Display");
    printf("\n3:search");
    printf("\n4:Exit");
    printf("\nEnter the Choice:");
    scanf("%d",&ch);
    switch(ch)
```

```
    {
            case 1: root=create();
                     break;
            case 3:printf("\nEnter the value to search:");
                     scanf("%d",&val);
                     resp=search(root,val);
                     if(resp==1)
                       printf("\nValue is found in Tree.");
                     else
                       printf("\nValue not found in Tree.");
                     break;
            case 2: display(root);
                     break;
            case 4: exit(0);
    }
  }
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
   struct NODE *prev;
   int data;
   struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node *getnode()
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->prev=NULL;
  temp->next=NULL;
  return temp;
}
void create()
{
    int n,i;
    node *temp;
    printf("\nEnter total nodes: ");
```

```
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
      temp=getnode();
      if(list==NULL)
            list=temp;
      else
       {
             last->next=temp;
             temp->prev=last;
       }
       last=temp;
    }
}

void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
    printf("->NULL");
}
void displayrev()
{
   node *ptr;
```

```
  for(ptr=last;ptr!=NULL;ptr=ptr->prev)
    printf("%d->",ptr->data);
  printf("->NULL");
}

void main()
{
  clrscr();
  create();
  printf("\nDoubly LL: ");
  display();
  printf("\nreverse Doubly LL: ");
  displayrev();
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;

int num;

node * getnodenum() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  temp->data=num;
  temp->next=NULL;
  return temp;
 }


node * create(node *list)
{
    node *temp,*last;
    temp=getnodenum();
```

```
        if(list==NULL)
                list=temp;
        else
        {
          for(last=list;last->next!=NULL;last=last->next);
           last->next=temp;
        }
        return list;

}
void display(node *list)
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("NULL");
}
void main()
{
   int n,i;
   node *positive=NULL,*negative=NULL;
   clrscr();
   printf("\nEnter total no to accept: ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
```

```
    {
      printf("\nEnter Positive or negative No.: ");
      scanf("%d",&num);
      if(num>0)
               positive=create(positive);
              else
                negative=create(negative);
    }
    printf("\nPostive Number List is: ");
    display(positive);
    printf("\nNegative Number List is : ");
    display(negative);
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct NODE *lchild;
  int data;
  struct NODE *rchild;
};
typedef struct NODE node;
node* getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nEnter the data: ");
   scanf("%d",&temp->data);
   temp->lchild=NULL;
   temp->rchild=NULL;
   return temp;
}

node *create()
{
  int ch,flag;
  char dir;
  node *root,*temp,*ptr;
```

```
root=NULL;
do{
   temp=getnode();
   flag=0;
   if(root==NULL)
         root=temp;
   else
   {
         ptr=root;
         while(!flag)
         {
           printf("\nCurrent Node is %d",ptr->data);
           printf("\nEnter thr Direction(L/R):");
           dir=getche();
           if(dir=='L' || dir=='l')
           {
             if(ptr->lchild==NULL)
             {
                    ptr->lchild=temp;
                    flag=1;
             }
             else
                   ptr=ptr->lchild;
           }
           else   if(dir=='R' || dir=='r')
```

```
            {
              if(ptr->rchild==NULL)
              {
                    ptr->rchild=temp;
                    flag=1;
              }
              else
                   ptr=ptr->rchild;
            }
          }//while
     }//else
     printf("\nDo you want to add more node(Y/N): ");
   //  scanf(" %c",&ch);
     ch=getche();
    }while(ch=='Y'||ch=='y');
    return root;
}
void postorder(node *ptr)
{
 if(ptr!=NULL)
 {
   postorder(ptr->lchild);
   postorder(ptr->rchild);
   printf("   %d",ptr->data);
 }
```

```
}
void preorder(node *ptr)
{
  if(ptr!=NULL)
  {
    printf("   %d",ptr->data);
    preorder(ptr->lchild);
    preorder(ptr->rchild);
  }
}
void inorder(node *ptr)
{
  if(ptr!=NULL)
  {
    printf("   %d",ptr->data);
    inorder(ptr->lchild);
    inorder(ptr->rchild);
  }
}

void main()
{
  int ch;
  node *root=NULL;
  clrscr();
```

```
  root=create();
  printf("\nIndorder Traversal: ");
  inorder(root);
  printf("\nPreorder Traversal:");
  preorder(root);
  printf("\nPostorder Traversal:");
  postorder(root);
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  struct node *prev;
  int data;
  struct NODE *next;
};
typedef struct NODE node;

node *list=NULL,*last;
node * getnode()
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  temp->prev=NULL;
  return temp;
}

void create()
{
  int n,i;
```

```
  node *temp;
  printf("\nEnter the no. of nodes: ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
    if(list==NULL)
            list=temp;
    else
     {
            last->next=temp;
            temp->prev=last;
     }
          last=temp;
          last->next=list;
          list->prev=last;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr->next!=list;ptr=ptr->next)
    printf("%d->",ptr->data);
  printf("%d->",ptr->data);
   printf("NULL");
```

```
}
void main()
{
  clrscr();
  create();
  printf("\nCircular Doubly LL:  u");
  display();
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;

node *front=NULL,*rear=NULL;
int isempty()
{
  if(front==NULL && rear==NULL)
    return 1;
  else
     return 0;
}
node * getnodenum(int data)
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   temp->data=data;
   temp->next=NULL;
   return temp;
}
```

```
void add(int data)
{
  node *temp;
  temp=getnodenum(data);
  if(front==NULL)
   {
        rear=temp;
        front=temp;
        rear->next=front;
   }
   else
    {
         rear->next=temp;
         rear=temp;
         rear->next=front;
    }
}

int deleteq()
{
 node *ptr;
 int val;
 ptr=front;
 val=front->data;
 if(front->next!=front)
```

```
  {
    front=front->next;
    rear->next=front;
  }
  else
  {
    front=NULL;
    rear=NULL;
  }
  free(ptr);
  return(val);
}
void display()
{
  node *ptr;
  for(ptr=front;ptr!=rear;ptr=ptr->next)
    printf(" %d",ptr->data);
  printf(" %d",ptr->data);
}

void main()
{
   int ch,data;
   clrscr();
   while(1)
```

```
{
  printf("\n1: Add.");
  printf("\n2: Delete");
  printf("\n3: Display.");
  printf("\n4: Exit");
  printf("\nEnter the Choice:");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:printf("\nEnter the data to insert in queue:");
              scanf("%d",&data);
              add(data);
              break;
    case 2: if(isempty())
                    printf("\nQueue is Empty!");
              else
                     printf("\nDeleted element is: %d",deleteq());
              break;
    case 3:
                    if(isempty())
                      printf("\nQueue is Empty!");
              else
                     display();
              break;
    case 4: exit(1);
```

```
    }
  }
  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nenter the data:");
   scanf("%d",&temp->data);
   temp->next=NULL;
   return temp;
}
void create()
{
   int i,n;
   node *temp;
   printf("\nEnter the total no of nodes");
   scanf("%d",&n);
   for(i=0;i<n;i++)
```

```
  {
        temp=getnode();
        if(list==NULL)
         list=temp;
        else
         last->next=temp;
        last=temp;
   }
}


void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
        printf("%d->",ptr->data);
  printf("NULL");
}
node * sortafter()
{
  node *ptr1,*ptr2,*temp;
  for(ptr1=list;ptr1!=NULL;ptr1=ptr1->next)
  {
    for(ptr2=ptr1->next;ptr2!=NULL;ptr2=ptr2->next)
    {
```

```
            if(ptr1->data>ptr2->data)
            {
              temp=ptr1->data;
              ptr1->data=ptr2->data;
              ptr2->data=temp;
            }
     }
  }
  return list;
}
void main()
{
   clrscr();
   create();
   display();
   list=sortafter();
   printf("\nLL after sorting: ");
   display();
   getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
```

```
  node *temp;
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
          if(list==NULL)
                  list=temp;
          else
            last->next=temp;
          last=temp;
  }
}
void display()
{
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);

void main()

{
   int ch;
   clrscr();
```

```
while(1)
{
        printf("\n1:Create the Linked list.");
        printf("\n2:Display the Linked List.");
        printf("\n3:Exit.");
        printf("\nEnter the choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
          case 1: create();
                  break;
          case 2: display();
                  break;
          case 3:exit();
        }
}   getch(); }
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode() //create a node
{
  node *temp;
  temp=(node *)malloc(sizeof(node));
  printf("\nEnter the data: ");
  scanf("%d",&temp->data);
  temp->next=NULL;
  return temp;
 }



void create()
{
  int n,i;
  node *temp;
```

```
  printf("\nEnter total nodes to be creted");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    temp=getnode();
          if(list==NULL)
                  list=temp;
          else
            last->next=temp;
          last=temp;
  }
}
void display()
{
            node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
    printf("\t%d",ptr->data);
}
node * search(int val)
{
  node *ptr;
  int pos;
   for(ptr=list,pos=1;ptr!=NULL && ptr->data!=val;ptr=ptr->next,pos++);
   return ptr;
```

```
}
void insertfirst()
{
    node *temp;
    temp=getnode();
    temp->next=list;
    list=temp;

}
void insertval(int val)
{
   node *ptr,*temp;
   ptr=search(val);
   if(ptr!=NULL)
    {
            temp=getnode();
            temp->next=ptr->next;
            ptr->next=temp;
    }

}

void insertpos(int pos)
{
  node *temp,*ptr;
```

```
 int cnt;
 if(pos==1)
 {
   temp=getnode();
  temp->next=list;
  list=temp;
 }
 else
 {
   for(ptr=list,cnt=1;ptr!=NULL && cnt<pos-1;ptr=ptr->next,cnt++);
   if(ptr!=NULL)
   {
          temp=getnode();
          temp->next=ptr->next;
          ptr->next=temp;
   }
   else
     printf("\nPosition is Invalid.");
     }
 }
node *deletefirst()
{
  node *ptr;
  ptr=list;
  list=ptr->next;
```

```
   free(ptr);
printf("\nnode is deleted");
   return list;
}

node * deletelast()
{
  node *ptr,*prev;
  for(ptr=list,prev=list;ptr->next!=NULL;prev=ptr,ptr=ptr->next);
    free(ptr);
    prev->next=NULL;
  return list;
}
node *deleteval()
{
   int val;
   node *ptr,*prev;
   printf("\nenter the val to delete the node: ");
   scanf("%d",&val);
   for(ptr=list,prev=list;ptr!=NULL && ptr->data!=val;prev=ptr,ptr=ptr->next);
   if(ptr!=NULL)
    {
      if(ptr==list && prev==list)
              list=ptr->next;
      else
```

```
            prev->next=ptr->next;
     free(ptr);
    }
    else
    printf("\ndata not found: ");
   return(list);
}

void main()
{
    int ch,val,pos,ch1;
    node *ptr;
    clrscr();

    while(1)
    {
            printf("\n1:Create the Linked list.");
            printf("\n2:Display the Linked List.");
            printf("\n3:Search .");
            printf("\n4: Insert at first Position.");
            printf("\n5:Insert in between by val.");
            printf("\n6:Insert by Position");
            printf("\n7:Delete.");
            printf("\n8:Exit.");
            printf("\nEnter the choice: ");
```

```
scanf("%d",&ch);
switch(ch)
{
  case 1: create();
             break;
  case 2: display();
             break;
             case 3: printf("\nEnter the values to be searched: ");
             scanf("%d",&val);
             ptr=search(val);
             if(ptr->data==val)
                        printf("Value is found in LL at position.");
            if(ptr==NULL)
                       printf("Value not found.");

             break;
  case 4:insertfirst();
            break;
  case 5:printf("\nEnter the val after which new data to be inserted: ");
             scanf("%d",&val);
             insertval(val);
             break;
  case 6:printf("\nenter the Position to insert the data: ");
            scanf("%d",&pos);
             insertpos(pos);
```

```
                    break;
            case 7: printf("\n71:delete first node.");
                    printf("\n72: delete last node.");
                    printf("\n73: delete node by val.");
                    printf("\n74: delete node by pos");
                    printf("\nenter the choice for delete option: ");
                    scanf("%d",&ch1);
                    switch(ch1)
                    {
                      case 71: list=deletefirst();
                               break;
                      case 72: list=deletelast();
                               break;
                      case 73: list=deleteval();
                               break;
                      case 74:
                               break;
                    }
                    break;
            case 8:exit();
                   }
    }
    getch();
}
```

```
#include<stdio.h>
#include<conio.h>
struct NODE
{
  int data;
  struct NODE *next;
};
typedef struct NODE node;
node *list=NULL,*last;
node * getnode()
{
   node *temp;
   temp=(node *)malloc(sizeof(node));
   printf("\nenter the data:");
   scanf("%d",&temp->data);
   temp->next=NULL;
   return temp;
}
void sortcreate()
{
   int i,n;
   node *temp,*ptr,*prev           ;
   printf("\nEnter the total no of nodes");
   scanf("%d",&n);
   for(i=0;i<n;i++)
```

```
  {
        temp=getnode();
        if(list==NULL)
          list=temp;
        else
        {
           for(ptr=list,prev=list;ptr!=NULL && temp->data>ptr->data;prev=ptr,ptr=ptr->next);
           if(ptr==list && prev==list)
           {
             temp->next=list;
             list=temp;
           }
           else
           {
             temp->next=prev->next;
             prev->next=temp;
           }
         }

   }
}



void display()
{
```

```
  node *ptr;
  for(ptr=list;ptr!=NULL;ptr=ptr->next)
        printf("%d->",ptr->data);
  printf("NULL");
}
void main()
{
  clrscr();
  sortcreate();
  display();

  getch();
}
```

```
#include<stdio.h>
#include<conio.h>
# define MAX 5
struct STACK
{
  int stk[MAX];
  int top;
};
typedef struct STACK stack;

//initialize the stack
void initstack(stack *s)
{
  int i;
  for(i=0;i<MAX;i++)
   s->stk[i]=0;
  s->top=-1;
}
int isempty(stack *s)
{
   if(s->top==-1)
     return 1;
   else
     return 0;
}
```

```
int isfull(stack *s)
{
  if(s->top==MAX-1)
    return 1;
  else
    return 0;
}
void push(stack *s,int data)
{
  s->top++;
  s->stk[s->top]=data;

}
int pop(stack *s)
{
  int val;
  val=s->stk[s->top];
  s->top--;
  return(val);  //return(s->stk[s->top--]);
}
void display(stack *s)
{
  int i;
  for(i=0;i<=s->top;i++)
     printf("\t%d",s->stk[i]);
```

```
}
void main()
{
   stack s;
   int ch,data;
   clrscr();
   initstack(&s);
   while(1)
   {
   printf("\nMain Menu.");
   printf("\n1:PUSH.");
   printf("\n2:POP.");
   printf("\n3:Display.");
   printf("\n4:Exit.");
   printf("\nEnter the Choice:");
   scanf("%d",&ch);
   switch(ch)
   {
    case 1:   if(isfull(&s))
                        printf("\nStack is FULL.");
                else
                {
                printf("\nEnter the data to PUSH:");
                scanf("%d",&data);
                push(&s,data);
```

```
                    }
                    break;
    case 2:   if(isempty(&s))
                        printf("\nStack is empty.");
                    else
                      printf("\nPopped data from stack is %d",pop(&s));
                    break;
    case 3:    if(isempty(&s))
                        printf("\nStack is empty.");
                    else
                    display(&s);
                    break;
    case 4: exit(0);
    }
   }
}
```

```
#include<stdio.h>
#include<conio.h>
struct student
{
   int rno;
   char sname[30];
   float per;
}*s[20];
void main()
{
  int n,i,j=0;
  clrscr();
  printf("\nEnter no of student to store in structure: ");
  scanf("%d",&n);
 for(i=0;i<n;i++)
 {
   s[i]=(struct student*)malloc(sizeof(struct student));
  printf("\nEnter rno: ");
  scanf("%d",&s[i]->rno);
  printf("\nEnter student name: ");
  scanf(" %s",&s[i]->sname);
 // printf("\nEnter percentage: ");
 // scanf("%f",&s->per);
 }
 for(j=0;j<n;j++)
```

```
 {
   printf("\nRno: %d\t",s[j]->rno);
   printf("name : %s ",s[j]->sname);

 }

 getch();
}
```