Department of Computer Science & Engineering
Faculty of Engineering & Technology

Data Structures Using C

PROGRAMMS IN DATA STRUCTURES

Suyash Bhardwaj

INDEX

Stack using array

```
#include<stdio.h>
#include<conio.h>
#define s 5
int top=-1,stk[s];
void main()
{
        intch=0;
        while(ch<4)
        {
                clrscr();
                printf("\n\t1. Push\n\t2. Pop\n\t3. List\n\t4. Exit\n\n\t  Enter your
choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                        push();
                        break;

                        case 2:
                        pop();
                        break;

                        case 3:
                        list();
                        break;
                }
        }
}
push()
{
        int a;
        if(top==s-1)
        {
                printf("\n\n\tstack is overflow");
                getch();
```

```
        return;
}
printf("\n\n\tenter data to push : ");
scanf("%d",&a);
top++;
stk[top]=a;
printf("\n\tdata is pushed");
        getch();
}

pop()
{
        if(top==-1)
        {
                printf("\n\tstack is underflow");
                getch();
                return;
        }
        printf("\n\n\tpopped data is : %d",stk[top]);
        top--;
        getch();
}

list()
{
        int i;
        if(top==-1)
        {
                printf("\n\n\tstack is underflow");
                getch();
                return;
}
printf("\n\n");
for(i=top;i>=0;i--)
{
        printf("\n\t%d",stk[i]);
}
```

```
        getch();
}
```

Queue using array

```
#include<stdio.h>
#define s 5
int rear=-1,front=-1,que[s];
main()
{
        intch=0;
        while(ch<4)
        {
                clrscr();
                printf("\n\t1. Insert\n\t2. Delete\n\t3. Display\n\t4. Exit\n\n\t  Enter
your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                        ins();
                        break;

                        case 2:
                        del();
                        break;

                        case 3:
                        list();
                        break;
                }
        }
}
                ins()
{
        int a;
        if(rear==s-1)
        {
                printf("\n\n\tqueue is full");
                getch();
                return;
```

```
}
printf("\n\n\tenter data to insert : ");
scanf("%d",&a);
rear++;
que[rear]=a;
if(front==-1)
{
        front++;
}
printf("\n\tdata is inserted");
getch();
}
del()
{
        if(rear==-1)
        {
                printf("\n\tqueue is empty");
                getch();
                return;
        }
        printf("\n\n\tdeleted data is : %d",que[front]);
        if(front==rear)
        {
                front=rear=-1;
        }
        else
        {
                front++;
        }
        getch();
}
list()
{
        int i;
        if(front==-1)
        {
                printf("\n\n\tqueue is empty");
```

```
            getch();
            return;
}
printf("\n\n");
for(i=front;i<=rear;i++)
{
            printf("  %d",que[i]);
            }
            getch();
}
```

Circular queue using array

```
#include<stdio.h>
#define max 3
int q[10],front=0,rear=-1;
void main()
{
intch;
   void insert();
   void delet();
   void display();
clrscr();
printf("\nCircular Queue operations\n");
printf("1.insert\n2.delete\n3.display\n4.exit\n");
   while(1)
   {
printf("Enter your choice:");
scanf("%d",&ch);
       switch(ch)
       {
       case 1: insert();
          break;
       case 2: delet();
          break;
       case 3:display();
          break;
       case 4:exit();
default:printf("Invalid option\n");
       }
   }
}
```

```
void insert()
{
int x;
   if((front==0&&rear==max-1)||(front>0&&rear==front-1))
printf("Queue is overflow\n");
   else
```

```
   {
printf("Enter element to be insert:");
scanf("%d",&x);
        if(rear==max-1&&front>0)
        {
            rear=0;
            q[rear]=x;
        }
        else
        {
            if((front==0&&rear==-1)||(rear!=front-1))
                q[++rear]=x;
        }
    }
}
void  delet()
{
int a;
    if((front==0)&&(rear==-1))
    {
printf("Queue is underflow\n");
getch();
        exit();
    }
    if(front==rear)
    {
        a=q[front];
        rear=-1;
        front=0;
    }
else
    if(front==max-1)
    {
        a=q[front];
        front=0;
    }
    else a=q[front++];
```

```
printf("Deleted element is:%d\n",a);
}

void display()
{
inti,j;
   if(front==0&&rear==-1)
   {
printf("Queue is underflow\n");
getch();
      exit();
   }
   if(front>rear)
   {
      for(i=0;i<=rear;i++)
printf("\t%d",q[i]);
      for(j=front;j<=max-1;j++)
printf("\t%d",q[j]);
printf("\nrear is at %d\n",q[rear]);
printf("\nfront is at %d\n",q[front]);
   }
   else
   {
      for(i=front;i<=rear;i++)
      {
printf("\t%d",q[i]);
      }
printf("\nrear is at %d\n",q[rear]);
printf("\nfront is at %d\n",q[front]);
   }
printf("\n");
}
getch();
```

Simple Link List

```
/*SINGLE LINKED LIST*/
#include<stdio.h>
#include<conio.h>
#include<alloc.h>

typedefstruct list
{
        int data;
        struct list *link;
}node;
node *first=NULL,*last=NULL;

display()
{
        clrscr();
        if(first==NULL)
        printf("\n\tNo data");
        else
        {
                node *temp=first;
                while(temp!=NULL)
                {
                        printf("%d\t",temp->data);
                        temp=temp->link;
                }
        }
}

insert()
{
        node *n,*cur=first,*prev=NULL;
        intch,len,pos,count=1;
        clrscr();
        if(first==NULL)
        {
                n=(node*)(malloc(sizeof(node)));
```

```
        printf("\nEnter first node..");
        scanf("%d",&n->data);
        n->link=NULL;
        first=last=n;
}
else
{
display();
printf("\n\n1.at beginning\n2.at end\n3.anywhere");
printf("\nEnter your choice..");
scanf("%d",&ch);
switch(ch)
{
        case 1:
        n=(node*)(malloc (sizeof(node)));
        printf("\nEnter data..");
        scanf("%d",&n->data);
        n->link=first;
        first=n;
        break;

        case 2:
        n=(node*)(malloc (sizeof(node)));
        printf("\nEnter data..");
        scanf("%d",&n->data);
        n->link=NULL;
        cur=first;
        prev=NULL;
        while(cur!=NULL)
        {
                prev=cur;
                cur=cur->link;
        }
        prev->link=n;
        last=n;
        break;
```

```
case 3:
printf("\nEnter position..");
scanf("%d",&pos);
len=0;
cur=first;
prev=NULL;
while(cur!=NULL)
{
        cur=cur->link;
        len++;
}
if(pos>len+1)
{
        printf("length = %d",len);
        printf("\nInvalid position");
        getch();
        break;
}
n=(node*)(malloc (sizeof(node)));
printf("\nEnter data..");
scanf("%d",&n->data);
if(pos==1)
{
        n->link=first;
        first=n;
}

else
{
        count=1;
        cur=first;
        prev=NULL;
        while(count<pos)
        {
                prev=cur;
                cur=cur->link;
                count++;
```

```
}
if(cur==NULL)
{
        prev->link=n;
        last=n;
}
else
        {
                prev->link=n;
                n->link=cur;
        }
        }//else
        break;
    }//switch
  }//else
  display();
}
delet()
{
  intch,i,val,pos,count=1,len=0;
  node *cur=first,*prev=NULL,*temp1=first;
  clrscr();
  display();
  if(first==NULL)
  {
        return;
  }
  printf("\n1.first node\n2.last node\n3.by value\n4.by position\n5.whole
list");
  printf("\nEnter your choice");
  scanf("%d",&ch);
  switch(ch)
  {
        case 1:
        first=first->link;
        break;
```

```
case 2:
while(cur->link!=NULL)
{
        prev=cur;
        cur=cur->link;
}
if(first==last)
first=last=NULL;
else
prev->link=NULL;
last=prev;
break;

case 3:
printf("\nEnter value");
scanf("%d",&val);
while(cur->data!=val&&cur!=NULL)
{
        prev=cur;
        cur=cur->link;
}
if(cur==NULL)
{
        printf("\nNot found");
        getch();
}
else if(cur==first)
first=first->link;
else if(cur==last)
{
        prev->link=NULL;
        last=prev;
}
else
{
        prev->link=cur->link;
        cur=cur->link;
```

```
}
break;

case 4:
printf("\nEnter position..");
scanf("%d",&pos);
while(temp1!=NULL)
{
        temp1=temp1->link;
        len++;
}
if(pos>len)
{
        printf("\nInvalid position");
        getch();
        break;
}
else
if(pos==1)
first=first->link;
else
{
        while(count<pos)
        {
                prev=cur;
                cur=cur->link;
                count++;
        }
        prev->link=cur->link;
        //cur=cur->link;
        }
        break;

        case 5:
        first=last=NULL;
        break;
} //switch
```

```
        display();
}


void main()
{
        intch;
        clrscr();
        display();
        while(1)
        {
                printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit");
                printf("\nEnter your choice..");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                        insert();
                        break;
                        case 2:
                        delet();
                        break;
                        case 3:
                        display();
                        break;
                        case 4:
                        exit(0);
                        default :printf("\nWrong choice");
                }
        }
}
```

Doubly Link List

```
/*DOUBLY LINKED LIST*/
#include<stdio.h>
#include<conio.h>
#include<alloc.h>

typedefstruct list
{
        int data;
        struct list *next;
        struct list *pre;
}node;
node *first=NULL,*last=NULL;

display()
{

        clrscr();
        if(first==NULL)
        printf("\nNo data");
        else
        {
                node *temp=first;
                while(temp!=NULL)
                {
                        printf("%d\t",temp->data);
                        temp=temp->next;
                }
        }
}
insert()
{
        node *n,*cur=first,*prev=NULL;
        intch,pos,count=1;
        clrscr();
        if(first==NULL)
        {
```

```
        n=(node*)(malloc (sizeof(node)));
        printf("\nEnter first node..");
        scanf("%d",&n->data);
        n->next=NULL;
        n->pre=NULL;
        first=last=n;
}
else
{
        node *temp1=first;intlen=1;
        display();
        printf("\n\n1.at beginning\n2.at end\n3.anywhere");
        printf("\nEnter your choice..");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:
                n=(node*)(malloc (sizeof(node)));
                printf("\nEnter data..");
                scanf("%d",&n->data);
                n->pre=NULL;
                n->next=first;
                first->pre=n;
                first=n;
                break;

                case 2 :
                n=(node*)(malloc (sizeof(node)));
                printf("\nEnter data..");
                scanf("%d",&n->data);
                n->next=NULL;
                n->pre=last;
                last->next=n;
                last=n;
                break;

                case 3:
```

```
printf("\nEnter position..");
scanf("%d",&pos);
while(temp1!=NULL)
{
        temp1=temp1->next;
        len++;
}
if(pos>len)
{
        printf("\nInvalid position");
        getch();
        break;
}
n=(node*)(malloc (sizeof(node)));
printf("\nEnter data..");
scanf("%d",&n->data);
if(pos==1)
{
        n->pre=NULL;
        n->next=first;
        first=n;
}
else
{
        while(count<pos)
        {
                prev=cur;
                cur=cur->next;
                count++;
        }
        if(cur==NULL)
        {
                prev->next=n;
                n->pre=prev;
                last=n;
        }
        else
```

```
                {
                        prev->next=n;
                        n->next=cur;
                        n->pre=prev;
                        cur->pre=n;
                }
}//else
                        break;
                }//switch
        }  //else
        display();
}


delet()
{
        intch,i,val,pos,count=1,len=0;
        node *cur=first,*prev=NULL,*temp1=first;
        clrscr();
        display();
        printf("\n1.first node\n2.last node\n3.by value\n4.by position\n5.whole
list");
        printf("\nEnter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:
                first=first->next;
                break;

                case 2:
                while(cur->next!=NULL)
                {
                        prev=cur;
                        cur=cur->next;
                }
                prev->next=NULL;
```

```
last=prev;
break;

case 3:
printf("\nEnter value");
scanf("%d",&val);
while(cur->data!=val&&cur!=NULL)
{
        prev=cur;
        cur=cur->next;
}
if(cur==NULL)
{
        printf("\nNot found");
        getch();
}
else if(cur==first)
first=first->next;
else if(cur==last)
{
        prev->next=NULL;
        last=prev;
}
else
{
        prev->next=cur->next;
         cur=cur->next;
         cur->pre=prev;
}
break;

case 4:
printf("\nEnter position..");
scanf("%d",&pos);
while(temp1!=NULL)
{
        temp1=temp1->next;
```

```
            len++;
}
if(pos>len)
{
            printf("\nInvalid position");
            getch();
            break;
}
else
if(pos==1)
first=first->next;
else
{
            while(count<pos)
            {
                        prev=cur;
                        cur=cur->next;
                        count++;
            }
                        prev->next=cur->next;
                        cur=cur->next;
                        cur->pre=prev;
            }
            break;

            case 5:
            first=last=NULL;
            break;
} //switch
display();
}


void main()
{
            intch;
            clrscr();
```

```
while(1)
{
        printf("\n1.Insert\n2.Delete\n3.exit");
        printf("\nEnter your choice..");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:
                insert();
                break;
                case 2:
                delet();
                break;
                case 3:
                exit(0);
                default:
                printf("\nWrong choice");
        }
}
```

Circular Link List

```
/*CIRCULAR LINKED LIST*/
#include<stdio.h>
#include<conio.h>
#include<alloc.h>

typedefstruct list
                { int data;
               struct list *link;
                }node;
node *first=NULL,*last=NULL;

void main()
{
intch;
clrscr();
 while(1)
 {
printf("\n1.Insert\n2.Delete\n3.exit");
printf("\nEnter your choice..");
scanf("%d",&ch);
  switch(ch)
  {
   case 1:insert();
          break;
   case 2:delet();
         break;
   case 3:exit(0);
   default :printf("\nWrong choice");

  }
 }
}
display()
{
clrscr();
  if(first==NULL)
```

```
        printf("\nNo data");
  else
  {
   node *temp=first;
   do
   {
printf("%d\t",temp->data);
    temp=temp->link;
   }while(temp!=first);
  }
 }
insert()
 {
  node *n,*cur=first,*prev=NULL;
intch,pos,count=1;
clrscr();
  if(first==NULL)
  {
   n=(node*)(malloc (sizeof(node)));
printf("\nEnter first node..");
scanf("%d",&n->data);
   n->link=NULL;
   first=last=n;
   last->link=first;
  }
  else
  {
   node *temp1=first;intlen=1;
   display();
printf("\n\n1.at beginning\n2.at end\n3.anywhere");
printf("\nEnter your choice..");
scanf("%d",&ch);
   switch(ch)
   {
    case 1:n=(node*)(malloc (sizeof(node)));
          printf("\nEnter data..");
          scanf("%d",&n->data);
```

```
        n->link=first;
         first=n;
         last->link=first;
         break;
case 2 :n=(node*)(malloc (sizeof(node)));
      printf("\nEnter data..");
      scanf("%d",&n->data);
      n->link=NULL;
      last->link=n;
            last=n;
            last->link=first;
            break;
case 3:    printf("\nEnter position..");
            scanf("%d",&pos);
         /*  while(temp1!=NULL)
                { temp1=temp1->link;
                   len++;
                }
           if(pos>len)
            { printf("\nInvalid position");
           getch();
            break;
           }*/
           n=(node*)(malloc (sizeof(node)));
           printf("\nEnter data..");
           scanf("%d",&n->data);

           if(pos==1)
            {n->link=first;
             first=n;
              last->link=first;
             }
            else
           {
           while(count<pos)
             {
            prev=cur;
```

```
      cur=cur->link;
      count++;
     }
    prev->link=n;
    n->link=cur;
  }
    break;
    }//switch
 // display();
  }  //else
 }
delet()
 {
intch,i,val,pos,count=1,len=0;
  node *cur=first,*prev=NULL,*temp1=first;
clrscr();
  display();
printf("\n1.first node\n2.last node\n3.by value\n4.by position\n5.whole list");
printf("\nEnter your choice");
scanf("%d",&ch);
  switch(ch)
  {
   case 1:first=first->link;
          last->link=first;
         break;
   case 2:while(cur!=last)
          {
         prev=cur;
           cur=cur->link;
          }
        prev->link=NULL;
         last=prev;
         last->link=first;
         break;
 case 3:printf("\nEnter value");
        scanf("%d",&val);
```

```
while(cur->data!=val&&cur->link!=first)
        {
              prev=cur;
              cur=cur->link;
        }
        if(cur->data!=val)
         {
printf("\nNot found");
getch();
     }
   else if(cur==first)
            {
             first=first->link;
             last->link=first;
             }
           else if(cur==last)
               {
           prev->link=NULL;
                last=prev;
                last->link=first;
                }
                  else
                  prev->link=cur->link;
                break;
case 4:printf("\nEnter position..");
              scanf("%d",&pos);
               /*  while(temp1!=NULL)
                  { temp1=temp1->link;
                    len++;
                  }
if(pos>len)
    { printf("\nInvalid position");
   getch();
    break;
    }
   else */
    if(pos==1)
```

```
  { first=first->link;
   last->link=first;
   }
else
{
 while(count<pos)
 {
                    prev=cur;
                    cur=cur->link;
                    count++;
                  }
                prev->link=cur->link;
                 }
          break;
  case 5:first=last=NULL;
          break;
    } //switch
 display();
}
```

Stack Using Link List

```
/* STACK USING LINK LIST*/
#include<stdio.h>
#include<conio.h>
#include<alloc.h>

typedefstruct node
{
        int data;
        struct node *link;
}stack;
stack *top=NULL;

void main()
{
        intch;
        while(1)
        {
                clrscr();
                display();
                printf("\nEnter your choice\n1.push\n2.pop\n3.exit.. ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:push();
                        break;
                        case 2:pop();
                        break;
                        case 3:exit(0);
                }
        }
}

push()
{
        stack *temp;
        temp=(stack*)(malloc (sizeof(stack)));
```

```
if(top==NULL)
{
        printf("\nEnter first node");
        scanf("%d",&temp->data);
        temp->link=NULL;
        top=temp;
}
        else
        {
                printf("\nEnter data");
                scanf("%d",&temp->data);
                temp->link=top;
                top=temp;
        }
        clrscr();
}

pop()
{
        if(top==NULL)
        printf("\nUnderflow");
        else
        {
                printf("\npoped element is %d ",top->data);
                top=top->link;
        }
        getch();
        clrscr();
}

display()
{
        stack *temp=top;
        if(top==NULL)
        printf("\nNO data");
        else
        {
```

```
        printf("\ntop-> ");
        while(temp!=NULL)
        {
                printf("%d\n",temp->data);
                temp=temp->link;
        }
}
```

Queue Using Link List

```
/****Program to Implement Queue using Linked List ****/
#include<stdio.h>
struct node
{
        int info;
        struct node *link;
}*front = NULL, *rear = NULL;

void insert();
void delet();
void display();
int item;
main()
{
        intch=0;
        while(ch<3)
        {
                clrscr();
display();
printf("\n\n1.\tInsert\n2.\tDelete\n3.\tExit\n");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
        case 1:
        insert();
        break;

        case 2:
        delet();
        break;

        case 3:
        exit(0);
        default:
        printf("\n\nInvalid choice. Please try again...\n");
```

```
                }
        }
        getch();
}
void insert()
{
        printf("\n\nEnter ITEM: ");
        scanf("%d", &item);
        if(rear == NULL)
        {
                rear = (struct node *)malloc(sizeof(struct node));
                rear->info = item;
                rear->link = NULL;
                front = rear;
        }
        else
        {
                rear->link = (struct node *)malloc(sizeof(struct node));
                rear = rear->link;
                rear->info = item;
                rear->link = NULL;
        }
}
void delet()
{

        struct node *ptr;
        if(front == NULL)
        printf("\n\nQueue is empty.\n");
        else
        {
                ptr = front;
                item = front->info;
                front = front->link;
                free(ptr);
                printf("\nItem deleted: %d\n", item);
                if(front == NULL)
```

```
                rear = NULL;
        }
}
void display()
{
        struct node *ptr = front;
        if(rear == NULL)
        printf("\n\nQueue is empty.\n");
        else
        {
                printf("\n\n");
                while(ptr != NULL)
                {
                        printf("%d\t",ptr->info);
                        ptr = ptr->link;
                }
        }
}
```

Insertion Sort

```
//insertion Sort//
#include<conio.h>
#include<stdio.h>
int main()
{
inti,j,s,temp,a[20];
clrscr();
printf("Enter total elements: ");
scanf("%d",&s);
printf("Enter %d elements: ",s);
for(i=0;i<s;i++)
scanf("%d",&a[i]);
for(i=1;i<s;i++)
{
        temp=a[i];
        j=i-1;
        while((temp<a[j])&&(j>=0))
        {
                a[j+1]=a[j];
                j=j-1;
        }
        a[j+1]=temp;
}
printf("After sorting: ");
for(i=0;i<s;i++)
printf(" %d",a[i]);
getch();
}
```

Selection Sort

```
/*SELECTION SORTING*/
#include<stdio.h>
#include<conio.h>
#define max 50
void main()
{
int a[max],n,i,order,temp,j;
clrscr();
printf("\nEnter number of elements");
scanf("%d",&n);
clrscr();
for(i=0;i<n;i++)
{
        printf("\nEnter element %d ",i+1);
        scanf("%d",&a[i]);
}
printf("\nEnter order (1-desc,0-asc)");
scanf("%d",&order);
if(order==0)
{
        for(i=0;i<n;i++)
                for(j=i+1;j<n;j++)
                        if(a[i]>a[j])
                        {
                                temp=a[i];
                                a[i]=a[j];
                                a[j]=temp;
                        }
}
else
{
        for(i=0;i<n;i++)
                for(j=i+1;j<n;j++)
                        if(a[i]<a[j])
                        {
                                temp=a[i];
```

```
                a[i]=a[j];
                a[j]=temp;
        }
}
printf("\nsorted array");
for(i=0;i<n;i++)
printf("\t%d",a[i]);
                getch();
}
```

Bubble Sort

```
/*BUBLE SORTING*/
#include<stdio.h>
#include<conio.h>
#define max 50
void main()
{
int a[max],n,i,order,temp,j;
clrscr();
printf("\nEnter number of elements");
scanf("%d",&n);
clrscr();
for(i=0;i<n;i++)
{
        printf("\nEnter element %d ",i+1);
        scanf("%d",&a[i]);
}
printf("\nEnter order (1-desc,0-asc)");
scanf("%d",&order);
if(order)
{
        for(i=0;i<n;i++)
                for(j=1;j<n-i;j++)
                        if(a[j]>a[j-1])
                        {
                                temp=a[j];
                                a[j]=a[j-1];
                                a[j-1]=temp;
                        }
}
else
{
        for(i=0;i<n;i++)
                for(j=1;j<n-i;j++)
                        if(a[j]<a[j-1])
                        {
                                temp=a[j];
```

```
                        a[j]=a[j-1];
                        a[j-1]=temp;
                    }
    }
    printf("\nsorted array");
    for(i=0;i<n;i++)
    printf("\t%d",a[i]);
    getch();
     }
```

Quick Sort

```
/* QUICK SORTING */
#include<stdio.h>
#include<conio.h>
#define max 50

int a[max],order;
void main()
{
intn,i,temp,j;
clrscr();
printf("\nEnter number of elements");
scanf("%d",&n);
clrscr();
 for(i=0;i<n;i++)
  {
printf("\nEnter element %d ",i+1);
scanf("%d",&a[i]);
  }
printf("Enter the order of sorting(0-asc,1-desc)");
scanf("%d",&order);
 quicksort(0,n-1);
 for(i=0;i<n;i++)
printf("\t%d",a[i]);
getch();
}

 quicksort(intlow,int high)
 {
int i;
 if(low<high)
 {
  i=partition(low,high);
  quicksort(low,i-1);
  quicksort(i+1,high);
 }
return;
```

```
 }

int partition(intlow,int high)
 {
intli,hi,x,temp;
  x=a[low];
  li=low+1;
hi=high;
if(order)
{
 while(1)
 {
  while(a[hi]<x)
     hi-=1;
  while(a[li]>=x)
     li+=1;
  if(li<hi)
  {
   temp=a[hi];
   a[hi]=a[li];
   a[li]=temp;
  }
  else
   {
   temp=a[hi];
   a[hi]=a[low];
   a[low]=temp;
   return hi;
  }
}
}
 else
   while(1)
   {
    while(a[hi]>x)
        hi-=1;
    while(a[li]<=x)
```

```
    li+=1;
  if(li<hi)
   {
    temp=a[hi];
    a[hi]=a[li];
    a[li]=temp;
   }
     else
      {
       temp=a[hi];
       a[hi]=a[low];
       a[low]=temp;
       return hi;
      }
   }
}
```

Sequential Search

```
/*Sequential SEARCH*/
#include<stdio.h>
#include<conio.h>
#define max 50
void main()
{
int a[max],n,i,item,pos,flag=0;
clrscr();
printf("\nEnter number of elements");
scanf("%d",&n);
clrscr();
 for(i=0;i<n;i++)
  {
printf("\nEnter element %d ",i+1);
scanf("%d",&a[i]);
 }
printf("Enter element you want to search..");
scanf("%d",&item);
 for(i=0;i<n;i++)
   if(a[i]==item)
      {
         pos=i+1;
         flag=1;
         break;
      }

 if(flag)
printf("\nElement found at %d position ",pos);
 else
printf("\nnot found");
getch();
}
```

Binary Search

```
//binary search//
#include<stdio.h>

int s;
void main()
{
int x[10],n,i;
intl,h,mid,f=0;
clrscr();
printf("\n\tenter total nos of elements");
scanf("%d",&s);
printf("\n\tenter all the elements");
for(i=0;i<s;i++)
{
        scanf("%d",&x[i]);
}
printf("\n\tenter the element to search");
scanf("%d",&n);

l=0,h=s-1;
while(l<=h)
{
        mid=(l+h)/2;
        if(x[mid]==n)
        {
                f=f+1;
                printf("\n\tdata found at location %d",mid);
                getch();
                return;
        }
if(x[mid]>n)
{
        h=mid-1;
}
else
{
```

```
                l=mid+1;
        }
}
if(f==0)
{
        printf("\n\tdata not found");
}
        getch();
}
```

Binary Search Tree

```
#include<stdio.h>
#include<alloc.h>
struct node
{
        int data;
        struct node *lc,*rc;
};

main()
{
        struct node *r=NULL;
        intch=0,a;
        while(ch<6)
        {
                clrscr();
                printf("\n\t1. Insert\n\t2. Preorder\n\t3. Inorder\n\t4. Postorder\n\t5.
Delete\n\t6. Exit\n\n\tEnter your choice : ");
                scanf("%d",&ch);
switch(ch)
{
        case 1:
        printf("\n\tenter data to insert : ");
        scanf("%d",&a);
        insert(&r,a);
        break;

        case 2:
        if(r==NULL)
        {
                printf("\n\tbtree is empty");
                getch();
}
else
{
                printf("\n\n\t");
                preorder(&r);
```

```
            getch();
}
break;

case 3:
if(r==NULL)
{
            printf("\n\tbtree is empty");
            getch();
}
else
{
            printf("\n\n\t");
            inorder(&r);
            getch();
}
break;

case 4:
if(r==NULL)
{
            printf("\n\tbtree is empty");
            getch();
}
else
{
            printf("\n\n\t");
            postorder(&r);
            getch();
}
break;

case 5:
if(r==NULL)
{
            printf("\n\tbtree is empty");
            getch();
```

```
}
else
{
        printf("\n\tenetr data to delete : ");
        scanf("%d",&a);
        del(&r,a);
}
                break;
            }
        }
}

insert(struct node **root,int n)
{
        struct node *p,*z,*t;
        t=malloc(sizeof(struct node));
        t->data=n;
        t->lc=t->rc=NULL;
        if(*root==NULL)
        {
                *root=t;
                return;
        }

        p=z=*root;
        while(z!=NULL)
        {
                if(n > z->data)
                {
                        p=z;
                        z=z->rc;
                }
                else
                {
                        p=z;
                        z=z->lc;
                }
```

```
}
if(n > p->data)
{
        p->rc=t;
}
else
{
        p->lc=t;
    }
}

preorder(struct node **root)
{
    if(*root!=NULL)
    {
        printf("%d  ",(*root)->data);
        preorder(&((*root)->lc));
        preorder(&((*root)->rc));
    }
}

inorder(struct node **root)
{
    if(*root!=NULL)
    {
        inorder(&((*root)->lc));
        printf("%d  ",(*root)->data);
        inorder(&((*root)->rc));
    }
}

postorder(struct node **root)
{
    if(*root!=NULL)
    {
        postorder(&((*root)->lc));
        postorder(&((*root)->rc));
```

```
                printf("%d  ",(*root)->data);
        }
}


del(struct node **root,int n)
{
        struct node *p,*io,*z;
        p=z=*root;
        while(z!=NULL)
        {
                if((*root)->data==n)
                {
                        printf("\n\n\tserached node is root cannot be delete");
                        getch();
                        return;
                }
                if(z->data==n)
{
                        printf("\n\tdata found : %d",z->data);
                        printf("\n\tparent data : %d",p->data);
                        if(z==p->lc)
                        {
                                printf("\n\tleft child");
                        }
                        else
                        {
                                printf("\n\tright child");
                        }
                        getch();
                        break;
}
                if(z->data > n)
                {
                        p=z;
```

```
            z=z->lc;
        }
        else
        {
            p=z;
            z=z->rc;
        }
        if(z==NULL)
        {
            printf("\n\tnode doesn't exist");
            getch();
            return;
        }

        if(z->lc==NULL && z->rc==NULL)    //******** LEAF NODE
        {
            if(z==p->lc)
            {
                p->lc=NULL;
            }
            else
            {
                p->rc=NULL;
            }
            free(z);
            return;
        }

        if(z->lc!=NULL && z->rc==NULL)    //******** NODE HAS LEFT
    CHILD
            {
                if(z==p->lc)
                {
                    p->lc=z->lc;
                }
                else
```

```
        {
                p->rc=z->lc;
        }
        z->lc=NULL;
        free(z);
        return;
}

        if(z->lc==NULL && z->rc!=NULL)    //******** NODE HAS RIGHT
CHILD
        {
                if(z==p->lc)
                {
                        p->lc=z->rc;
                }
                else
                {
                        p->rc=z->rc;
                }
                z->rc=NULL;
                free(z);
                return;
        }

        if(z->lc!=NULL && z->rc!=NULL)    //******** NODE HAS BOTH
CHILD
        {
                p=z;
                io=z->rc;
                while(io->lc!=NULL)
                {
                        p=io;
                        io=io->lc;
                }
                z->data=io->data;

                if(io->rc==NULL)
```

```
{
        if(io==p->lc)
        {
                p->lc=NULL;
        }
        else
        {
                p->rc=NULL;
        }
        free(io);
        return;
}
else
{
        if(io==p->lc)
        {
                p->lc=io->rc;
        }
        else
        {
                p->rc=io->rc;
        }
        io->rc=NULL;
        free(io);
        return;
                }
        }
}
```