#include<iostream>

#include<bits/stdc++.h>

using namespace std;

int size=0;

class node

{

public:

int data;

node \*next;

node \*prev;

};

class LinkedList

{

public:

node \*head=NULL;

node \*tail;

void insert(int);

void insert(int,int);

void print();

void printDS();

void del(int);

void replac(int,int);

void inSort();

node \*Bsearch(int);

void swpPOS(int,int);

node \* Midptr(node\*,node\*);

};

void LinkedList::insert(int Data, int pos)

{

node \*temp1 = new node;

temp1->data = Data;

node \*temp2;

node \*temp;

temp=head;

if (pos == 1)

{

temp1->next = head;

head = temp1;

if (tail == NULL)

{

tail = temp1;

}

size++;

//print();

return;

}

else if ((pos < 1) || (pos > size))

{

cout << "Invalid request\n";

//print();

return;

}

else if (pos == size)

{

cout << pos << " " << size << "\n";

tail->next = temp1;

temp1->next = NULL;

tail = temp1;

size++;

//print();

return;

}

else

{

while(pos>2)

{

temp=temp->next;

pos--;

}

temp2=temp;

temp2=temp->next;

temp->next=temp1;

temp1->next=temp2;

temp1->prev=temp;

temp2->prev=temp1;

}

}

void LinkedList::insert(int Data)

{

node \*temp= new node;

temp->data=Data;

if(head==NULL)

{

head = temp;

tail = temp;

temp->next=NULL;

temp->prev=NULL;

size++;

return;

}

temp->next=NULL;

temp->prev=tail;

tail->next=temp;

tail=temp;

size++;

return;

}

void LinkedList::print()

{

if(head==NULL)

{

cout<<"Empty"<<endl;

}

else

{

node \*temp=head;

while(temp->next!=NULL)

{

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<temp->data<<endl;

}

}

void LinkedList::printDS()

{

if(head==NULL)

{

cout<<"Empty"<<endl;

}

else

{

node \*temp=tail;

while(temp->prev!=NULL)

{

cout<<temp->data<<" ";

temp=temp->prev;

}

cout<<temp->data<<endl;

}

}

void LinkedList::del(int pos)

{

node \*temp=head;

node \*temp1;

node \*temp2;

if(pos==1)

{

if(temp->next!=NULL)

{

temp1=temp->next;

delete temp;

head=temp1;

head->prev=NULL;

}

else

{

delete temp;

head=NULL;

}

}

else if(pos==size)

{

temp=tail->prev;

delete tail;

tail=temp;

temp->next=NULL;

size--;

}

else

{

pos--;

while(pos>1)

{

temp=temp->next;

pos--;

}

temp1=temp;

temp1=temp1->next;

temp2=temp1->next;

delete temp1;

temp1=NULL;

temp->next=temp2;

temp2->prev=temp;

}

}

void LinkedList::replac(int Data,int pos=size)

{

node \*temp=head;

node \*temp1;

node \*temp2;

if(pos==1)

head->data=Data;

else if(pos==size)

{

tail->data=Data;

}

else

{

while(pos>0)

{

temp=temp->next;

pos--;

}

temp->data=Data;

}

}

void LinkedList::inSort()

{

if(head==NULL)

{

cout<<"List is Empty!!"<<endl;

return;

}

node \* running=head;

node \* running2=head;

while(running->next!=NULL)

{

running2=running->next;

//cout<<running << " " << running->data <<" " << (running2)->data <<endl;

if((running->data)<=(running2->data))

{

running=running->next;

continue;

}

else if((head->data)>(running2->data))

{

node \*ntemp = new node;

ntemp->data=running2->data;

ntemp->next=head;

head->prev=ntemp;

head=ntemp;

head->prev=NULL;

if(running2->next!=NULL)

{

node \*tdel=running2;

running2=running2->next;

running->next=running2;

running2->prev=running;

tdel = NULL;

delete tdel;

continue;

}

else

{

node \*tdel=running2;

running->next=NULL;

tail=running;

tdel = NULL;

delete tdel;

return;

}

}

else if((running->data)>(running2->data))

{

node \* htemp=running;

node \* rtemp=running2;

node \*ntemp = new node;

ntemp->data=running2->data;

if(running2->next != NULL)

{

node \*tdel=running2;

running2=running2->next;

running->next=running2;

running2->prev=running;

tdel = NULL;

delete tdel;

}

else if(running2->next == NULL)

{

node \*tdel=running2;

running->next=NULL;

tail=running;

tdel = NULL;

delete tdel;

}

node \* Firstptr = head;

node \* Lastptr = running;

node \* middle;

do

{

middle = Midptr( Firstptr , Lastptr );

if( middle->data >= ntemp->data )

{

htemp=middle;

}

else if ( middle->data < ntemp->data )

{

Firstptr = middle->next;

}

else

{

Lastptr = middle;

}

}

while( Lastptr->data > Firstptr->data);

htemp=middle;

node \* temp;

temp=htemp->next;

ntemp->prev=htemp;

ntemp->next=temp;

htemp->next=ntemp;

temp->prev=ntemp;

}

}

tail=running;

tail->next=NULL;

}

void LinkedList::swpPOS(int a, int b)

{

int c,d;

node \* temp1 = head;

node \* temp2 = head;

while(a>1)

{

temp1=temp1->next;

a--;

}

c=temp1->data;

while(b>1)

{

temp2=temp2->next;

b--;

}

d=temp2->data;

temp1->data=d;

temp2->data=c;

return;

}

node \* LinkedList::Midptr( node \* Firstptr , node \* Lastptr)

{

if( Firstptr == NULL )

{

cout<<"Linked list is empty !!"<<endl;

return NULL;

}

node \* sjump = Firstptr;

node \* djump = Firstptr -> next;

while ( djump != Lastptr )

{

djump = djump -> next ;

if( djump != Lastptr )

{

sjump = sjump -> next ;

djump = djump -> next ;

}

}

return sjump ;

}

node \*LinkedList::Bsearch( int valueToSearch )

{

node \* Firstptr = head;

node \* Lastptr = NULL;

do

{

node \* middle = Midptr( Firstptr , Lastptr );

if( middle == NULL )

{

return NULL;

}

if( middle->data == valueToSearch )

{

return middle;

}

else if ( middle->data < valueToSearch )

{

Firstptr = middle->next;

}

else

{

Lastptr = middle;

}

}

while( Lastptr == NULL || Lastptr->next != Firstptr );

return NULL;

}

int main()

{

LinkedList ob;

int a,p;

ob.insert(1);

ob.insert(3);

ob.insert(2);

ob.insert(4);

ob.insert(5);

ob.insert(6);

ob.insert(7);

ob.insert(8);

cout<<"Display all : "<<endl;

ob.print();

//cout<<"\n\nAfter insert 109 int the fourth position: "<<endl;

// ob.insert(109,4);

// ob.print();

// cout<<"\n\nAfter replace the value 97 to the last position: "<<endl;

// ob.replac(97);

// ob.print();

// cout<<"\n\nAfter Deleting 1st Node: "<<endl;

// ob.del(1);

// ob.print();

ob.inSort();

cout<<"After Sorting: "<<endl;

ob.print();

//ob.Bsearch(17);

// cout<<ob.Bsearch(17)->next->data<<endl;

}