**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Programs** | **PAGE** | **REMARK** |
| 1 | Array using pointer | 1 |  |
| 2 | Stack using array | 1-4 |  |
| 3 | Stack using linked list | 4-6 |  |
| 4 | Decimal to binary | 6-8 |  |
| 5 | Balanced parentheses using stack | 8-9 |  |
| 6 | Queue using array | 9-12 |  |
| 7 | Queue using linked list | 12-14 |  |
| 8 | Circular Queue using array | 15-17 |  |
| 9 | Linear search | 17-18 |  |
| 10 | Binary search | 18-19 |  |
| 11 | Selection sort | 19-21 |  |
| 12 | Bubble sort | 21-22 |  |
| 13 | Merge sort | 22-21 |  |
| 14 | Insertion sort | 24-25 |  |
| 15 | Infix to postfix | 25-27 |  |
| 16 | Evaluate postfix expression using stack | 28-29 |  |
| 17 | Circular Queue using linked list | 29-32 |  |
| 18 | Single linked list | 32-35 |  |
| 19 | Doubly linked list | 36-39 |  |
| 20 | Binary search tree | 40-42 |  |
| 21 | Quick sort | 43-44 |  |
| 22 | Single Circular linked list | 44-46 |  |
| 23 | Doubly Circular linked list | 46-48 |  |
| 24 | Min heap | 49-51 |  |
| 25 | Max heap | 51-56 |  |
| 26 | Heap sort | 56-59 |  |

1. **Array using pointer**

#include <stdio.h>

Int main( )

{

int data[100], n,i;

printf("\nenter the size of array");

scanf("%d",&n);

printf("Enter elements: ");

for(i = 0; i < n; ++i)

scanf(" %d", data + i);

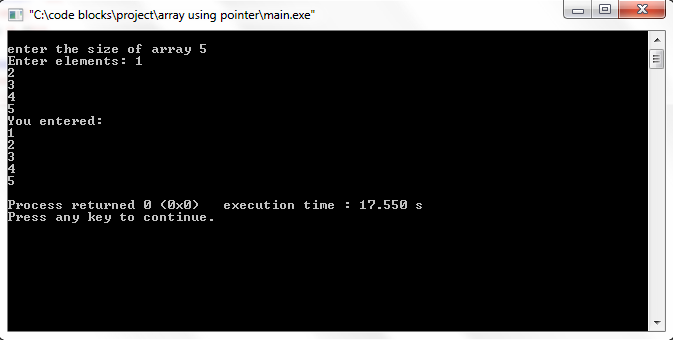
printf("\nYou entered:");

for(i = 0; i < n; ++i)

printf(" %d", \*(data + i));

return 0;

}



1. **Stack using array**

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 100

int stack[MAXSIZE],ch,n,x,top,i;

void push(void);

void pop(void);

void display(void);

void push()

{

if(top>=n-1)

{

printf("\nstack underflow");

}

else

{

printf("\nenter the value to be pushed: ");

scanf("%d",&x);

top++;

stack[top]=x;

}

}

void pop()

{

if(top==-1)

{

printf("\nstack underflow");

}

else

{

printf("\nthe poped element is : %d",stack[top]);

top--;

}

}

void display()

{

if(top>=0)

{

printf("\nthe stack element is ");

for(i=0;i<=top;i++)

{

printf(" %d",stack[i]);

}}

else

{printf("\n the stck is empty");

}

}

int main()

{

top=-1;

printf("\n enter the size of stack: ");

scanf("%d",&n);

do{

printf("\n enter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:push(); break;

case 2:pop(); break;

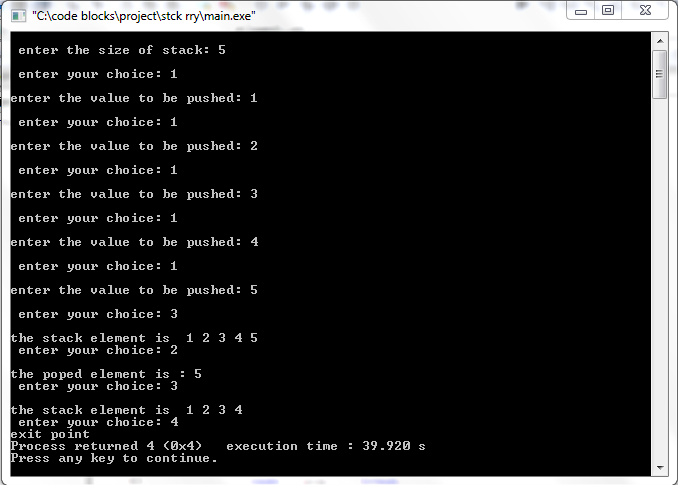
case 3:display(); break;

case 4: printf("exit point"); break;

}

}while(ch!=4);

}



1. **stack using linked list**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int dt;

struct node\* link;

};

struct node\* top=NULL;

void push()

{

struct node\* temp;

temp=(struct node\*)malloc(sizeof(struct node));

printf("\nenter your data : ");

scanf("\n%d",&temp->dt);

temp->link=top;

top=temp;

}

void pop()

{

struct node\* temp;

if(top==NULL)

{printf("\n stack underflow");}

else

{

temp=top;

printf(" \npoped element : %d", temp->dt);

top=top->link;

temp->link=NULL;

free(temp);

}

}

void display()

{

struct node\* temp;

if(top==NULL)

{printf("\n stack underflow");}

else

{

temp=top;

printf("\n Display : ");

while(temp!= NULL)

{

printf(" %d ",temp->dt);

temp=temp->link;

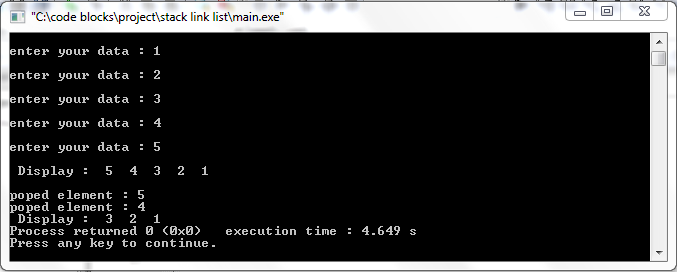
}

}

}

int main()

{ push();push();push();push();push();display();printf("\n");pop();pop();display();}



1. **Decimal to binary**

#include <stdio.h>

#include <stdlib.h>

#define m 30

void push(int\*,int\*,int\*);

int pop(int\*,int\*);

int main()

{

char ans='y';

int d,bin[m]={},i=0,j,p;

while(ans=='y')

{

printf("enter decimal value : ");

scanf("%d",&d);

while(d!=0)

{

push(bin,&i,&d);

}

printf("\nbinary value ");

for(j=i-1;j>=0;j--)

{

p=pop(&bin,&j);

printf("%d",p);

}

printf("\n");

printf("wants to be continue?(y/n)");

scanf("%c",&ans);

}

}

void push(int \*bin,int \*i,int \*d)

{

bin[\*i]=(\*d)%2;

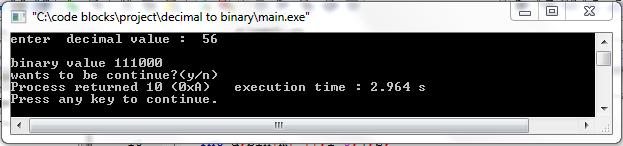
\*i=\*i+1;

\*d=(\*d)/2;

}

int pop(int \*bin,int \*i)

{ return(bin[\*i]);}



1. **Balanced parentheses using stack**

#include <stdio.h>

#include <stdlib.h>

int top=-1;

char st[100];

void push(char);

void pop();

void push(char x)

{ top++;

st[top]=x;}

void pop()

{

if(top==-1)

{printf("\n expression is invlid"); }

else

{ top--;}

}

int main()

{

int i;

char x[100];

printf("enter the expression : ");

scanf("%s",&x);

for(i=0;x[i]!='\0';i++)

{

if(x[i]=='(')

{push(x[i]);}

else

{ pop();}

}

check();

}

void check()

{

if(top==-1)

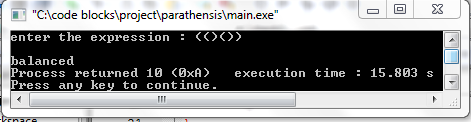
{

printf("\nbalanced ");

}else

{printf("\n unbalanced") }

}



1. **Queue using array**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

int queue[SIZE],f=-1,d,i,t,r=-1,cp=SIZE;

void enqueue(int d)

{

if(r==SIZE)

{

printf("\nfull");

}

else

{

r++;

queue[r]=d;

}

}

void dequeue()

{

if(f==r || f>r)

{

printf("\nempty");

}

else

{f++;

printf("\n the poped element is : %d",queue[f]);

}

}

void display()

{

if(f==r || f>r)

{

printf("\nempty");

}

else

{t=f;t=t+1;

printf("\n display");

for(i=t;i<=r;i++)

printf(" %d",queue[i]);

}

}

int main()

{

/\* enqueue(5);enqueue(6);enqueue(9);display();dequeue();dequeue();display();\*/

int ch,x;

printf("\n\*\*\*\*\*\*\*\*\*\*\* queue implement on using array \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n1.insert \n2.delete\n3.display\n4.exit\n");

while(1)

{ printf("\nenter your chioce: ");

scanf("%d",&ch);

switch(ch)

{case 1:printf("\n enter the value to insert : ");

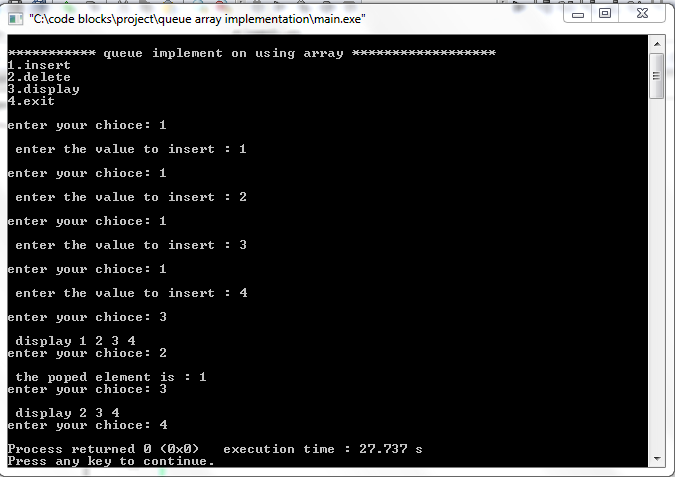
scanf("%d",&x);enqueue(x); break;

case 2:dequeue(); break;

case 3:display(); break;

case 4: exit(0);

default: printf("\n wrong selection please try again");}}}



1. **Queue using linked list**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

struct Node\* f=NULL;

struct Node\* r=NULL;

void enqueue(int x)

{

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node\*));

temp->data=x;

temp->next=NULL;

if(f==NULL & r==NULL)

{f=r=temp;}

else

{r->next=temp;

r=temp;}

}

void dequeue()

{

if(f==r)

{

printf("queue is empty");

}

else{

struct Node\*temp=f;

f=f->next;

printf("\ndeleted element: %d",temp->data);

free(temp);

}}

void display()

{ if(f==r)

{

printf("queue is empty");

}

else

{struct Node\*temp=f;

printf("\n display : ");

while(temp->next!=NULL)

{printf("%d ",temp->data);

temp=temp->next; }

printf("%d \n",temp->data);

} }

int main()

{

int ch,x;

printf("\n\*\*\*\*\*\*\*\*\*\*\*queue implement on using linked list\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n1.insert \n2.delete\n3.display\n4.exit\n");

while(1)

{

printf("\nenter your chioce: ");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("\n enter the value to insert : ");

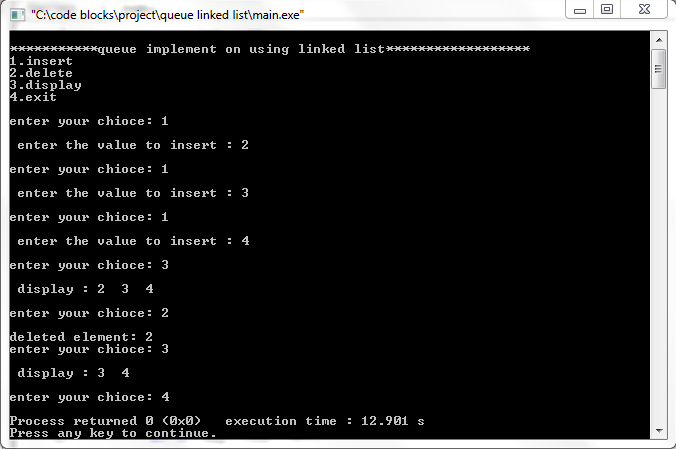
scanf("%d",&x);enqueue(x); break;

case 2:dequeue(); break;

case 3:display(); break;

case 4: exit(0);

default: printf("\n wrong selection please try again"); } } }



1. **Circular queue using array**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 5

int item[SIZE],f=-1,r=-1;

void enqueue(int x)

{

if(f==0 && r==SIZE-1 || f==r+1)

{

printf("\n queue is full");

}else if(f==-1)

{

f=0;}

r=(r+1)%SIZE;

item[r]=x;

printf("\n inserted element %d ",x);

}

void dequeue()

{

int c;

if(f==-1)

{

printf("\n queue is empty");

}else

{

c=item[f];

if(f==r)

{

f=-1;r=-1;

}

else{f=(f+1)%SIZE;}

/\*return(c);\*/

}

}

void display()

{ int i;

if(f==-1)

{

printf("\n queue is empty");

}

else

{ printf("\nfront-> %d",f);

printf("\nitem->");

for(i=f;(i)!=r;i=(i+1)%SIZE)

{printf("%d",item[i]);}

printf("%d",item[i]);

printf("\nrear-> %d \n",r);

}

}

int main()

{ int c;

enqueue(1);enqueue(2);enqueue(3);enqueue(4);enqueue(5); printf("\n");display();

dequeue();

/\*

if(c!=-1)

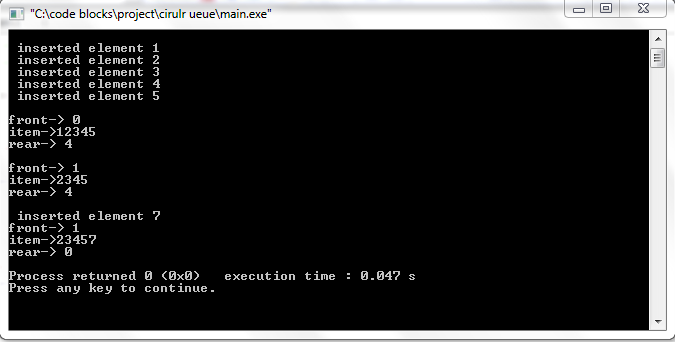
printf("\ndeleted element is -> %d",c);\*/

display();

enqueue(7);

display();

return 0; }



1. **linear search**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x[100],search,i,n;

printf("enter the size of array: ");

scanf("%d",&n);

printf("\nenter the elements :");

for(i=0;i<n;i++)

{

scanf(" \n %d",&x[i]);

}

printf("\n enter the search element:");

scanf("%d",&search);

for(i=0;i<n;i++)

{

if(x[i]==search)

{

printf("\n %d is present t location %d \n",search,i+1);

break;

}

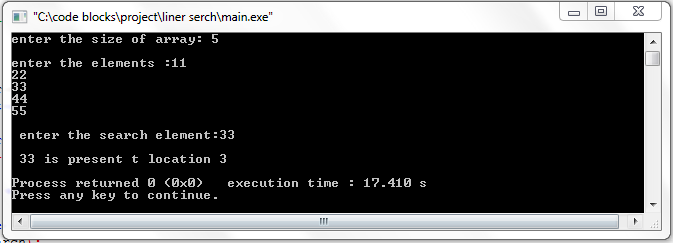
}

if(i==n)

{printf("\n element is not found ");}

return 0;

}



1. **Binary search**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x[100],search,i,n,first,last,mid;

printf("enter the size of array: ");

scanf("%d",&n);

printf("\nenter the elements :");

for(i=0;i<n;i++)

{scanf(" \n %d",&x[i]);}

printf("\n enter the search element:");

scanf("%d",&search);

first=0;

last=n-1;

mid=(first+last)/2;

while(first<=last)

{

if(x[mid]<search)

{ first=mid+1;}

else if(x[mid]==search)

{ printf("\n %d is present at location %d \n",search,mid+1); break; }

else

last=mid-1;

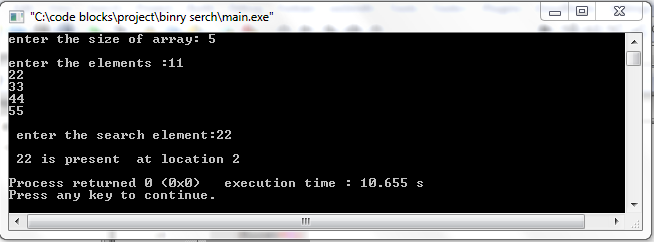
mid=(first+last)/2;

}

if(first>last)

{ printf("\n element is not found"); }

return 0;}



1. **Selection sort**

#include <stdio.h>

#include <stdlib.h>

void selection\_sort(int x[],int n)

{

int i,j,m;

for(i=0;i<n-1;i++)

{ int temp;

m=i;

for(j=i+1;j<n;j++)

{if(x[j]<x[m])

m=j;}

temp=x[m];

x[m]=x[i];

x[i]=temp;

}

}

void print(int x[] ,int s)

{

int i;

for(i=0;i<s;i++)

{printf("%d ",x[i]); }

printf("\n"); }

int main()

{

int x[]={64,25,12,22,11};

int n=sizeof(x)/sizeof(x[0]);

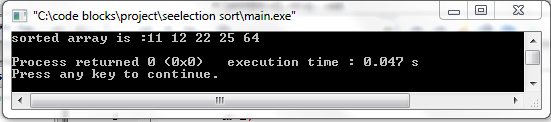
selection\_sort(x,n);

printf("sorted array is :");

print(x,n);

return 0;

}



1. **Bubble sort**

#include <stdio.h>

#include <stdlib.h>

void bubble\_sort(int x[],int n)

{

int i,j;

for(i=n-2;i>=0;i--)

{ int temp;

for(j=0;j<i;j++)

{if(x[j]>x[j+1])

temp=x[j];

x[j]=x[j+1];

x[j+1]=temp;

}}}

void print(int x[] ,int s)

{

int i;

for(i=0;i<s;i++)

{ printf("%d ",x[i]);}

printf("\n");

}

int main()

{

int x[]={64,34,25,12,22,11,90};

int n=sizeof(x)/sizeof(x[0]);

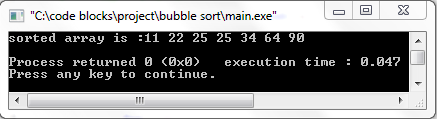
bubble\_sort(x,n);

printf("sorted array is :");

print(x,n);

return 0;

}



1. **Merge sort**

#include <stdio.h>

#include <stdlib.h>

#define mx 10

int x[11]={10,14,19,26,27,31,33,35,42,44,50};

int b[10];

void merging(int l,int m ,int h)

{

int l1,l2,i;

for(l1=l,l2=m+1,i=l;l1<m && l2<=h ; i++)

{

if(x[l1]<=x[l2])

b[i]=x[l1++];

else

b[i]=x[l2++];

}

while(l1<m)

b[i++]=x[l1++];

while(l2<=h)

b[i++]=x[l2++];

for(i=l;i<=h;i++)

x[i]=b[i];

}

void sort(int l,int h)

{

int m;

if(l<h)

{

m=(l+h)/2;

sort(l,m);

sort(m+1,h);

merging(l,m,h);

}}

int main()

{

int i;

printf("\nlist before sorting :");

for(i=0;i<=mx;i++)

{printf(" %d",x[i]);}

sort(0,mx);

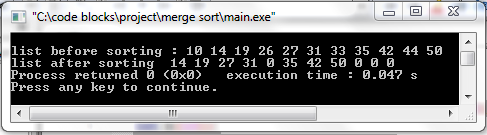
printf("\nlist after sorting ");

for(i=0;i<=mx;i++)

{printf(" %d",x[i]);}

return 0;

}



1. **Insertion sort**

#include <stdio.h>

#include <stdlib.h>

void insertion\_sort(int x[],int n)

{

int i,j,m;

for(i=1;i<n;i++)

{

m=x[i];

j=i-1;

while((j>=0)&&(x[j]>m))

{

x[j+1]=x[j];

j=j-1;

}

x[j+1]=m;

}

}

void print(int x[] ,int s)

{

int i;

for(i=0;i<s;i++)

{printf("%d ",x[i]);}

printf("\n");

}

int main()

{

int x[]={64,25,12,22,11};

int n=sizeof(x)/sizeof(x[0]);

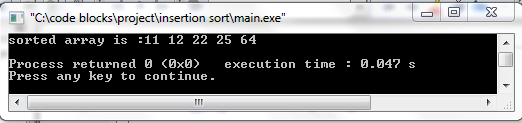
insertion\_sort(x,n);

printf("sorted array is :");

print(x,n);

return 0;

}

]

**15.Infix to postfix**

#include<stdio.h>

#include<ctype.h>

#include<string.h>

int priority(char);

int push(char);

int pop();

int top=0 ,s,p=0;

char stack[80],postfix[80];

int priority(char v)

{ if(v=='+' || v=='-')

return 1;

else if(v=='/' || v=='\*')

return 2;

else

return 0;

}

int push(char ch)

{ intj,k;

char z;

stack[0]='(';

j=priority(ch);

k=priority(stack[top]);

if(top==s)

printf("stack is full");

else

{ if(j>k)

{ stack[++top]=ch;

}

else{

if(ch=='(')

stack[++top]=ch;

else{

z=pop();

postfix[p++]=z;

stack[++top]=ch; }

} }

return 0;

}

int pop()

{ return stack[top--]; }

int main()

{ int c;

char infix[80];

printf("please enter your expression :\t");

scanf("%s",&infix);

s=strlen(infix);

infix[s]=')';

infix[s+1]='\0';

for (int i=0;i<=s;i++)

{

if(infix[i]=='(')

push(infix[i]);

else if(infix[i]=='+' || infix[i]=='/' || infix[i]=='-' || infix[i]=='\*' || infix[i]=='^')

push(infix[i]);

else if(infix[i]==')')

{ char x;

int k=1;

while(k)

{ x=pop();

if(x=='(')

k=0;

else

postfix[p++]=x; }

}

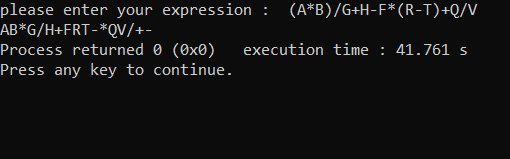
else

postfix[p++]=infix[i]; }

printf("%s",postfix);

return 0;

}

******

**16.Evaluate postfix expression using stack**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

struct node{

int top;

int capacity;

int \*array;

};

struct node \*createnode(int capacity)

{ struct node \*n;

n=(struct node\*)malloc(sizeof(struct node));

n->top=-1;

n->capacity=capacity;

n->array=(int \*)malloc(sizeof(int)\*capacity);

return n;

}

void push(struct node \*stack,int data)

{ stack->array[++stack->top]=data;

}

int pop(struct node \*stack)

{ if(stack->top==-1)

printf("Empty");

else{

return stack->array[stack->top--]; }

}

int main()

{ struct node \*stack;

char postfix[40];

int i,result,val1,val2;

printf("Enter the expression: ");

scanf("%s",&postfix);

stack=createnode(strlen(postfix));

for (i=0;postfix[i];i++)

{ if(isdigit(postfix[i]))

push(stack,postfix[i]-'0');

else{ val1=pop(stack);

val2=pop(stack);

switch(postfix[i])

{case '+':result= val2+val1;

push(stack,result);break;

case '-':result= val2-val1;

push(stack,result);break;

case '/':result= val2/val1;

push(stack,result);break;

case '\*':result= val2\*val1;

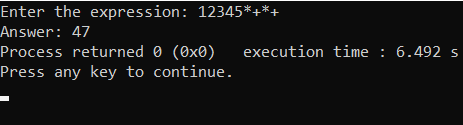
push(stack,result);break;

} }

}

printf("Answer: %d",pop(stack));

}

******

**17.circular queue using linked list**

#include<stdio.h>

#include<stdlib.h>

typedefstructcqueue{

int info;

structcqueue \*next;

}node;

node \*front=NULL, \*rear=NULL , \*temp;

intch;

voidenqueue()

{ node \*newnode;

newnode=(node\*)malloc(sizeof(node));

printf("Enter the element to be inserted in queue ");

scanf("%d",&newnode->info);

newnode->next=NULL;

if(rear==NULL)

{front=rear=newnode; }

else{ rear->next=newnode;

rear=newnode; }

}

voiddequeue()

{ temp=front;

if(front ==NULL)

{ printf("Queue is empty\n"); }

else{ if(front==rear){

printf("Popped element is: %d\n", front->info);

front=rear=NULL; }

else{printf("Popped element is:%d\n",front->info);

front=front->next;

rear->next=front; }

temp->next=NULL;

free(temp); }

}

void display(){

temp=front;

if(front==NULL){

printf("Queue is empty\n"); }

else{printf("\n");

for(;temp!=rear;temp=temp->next)

printf("\n%d",temp->info);

printf("\n%d",temp->info); }

}

void top(){

if(front==NULL){

printf("Queue is empty\n"); }

else{printf("Element at top is:%d\n",front->info); }

}

int main()

{ printf("\n \t\t----Welcome to the queue implementation program----\n");

printf("Choose from the operations to perform\n ");

printf("1.Enqueue\n 2.Dequeue\n 3.Top element\n 4.Print elements in the queue\n 5.Exit program\n ");

do{printf("\nEnter the choice\t");

scanf("%d",&ch);

switch(ch){

case 1: enqueue(); break;

case 2: dequeue(); break;

case 3: top(); break;

case 4: display(); break;

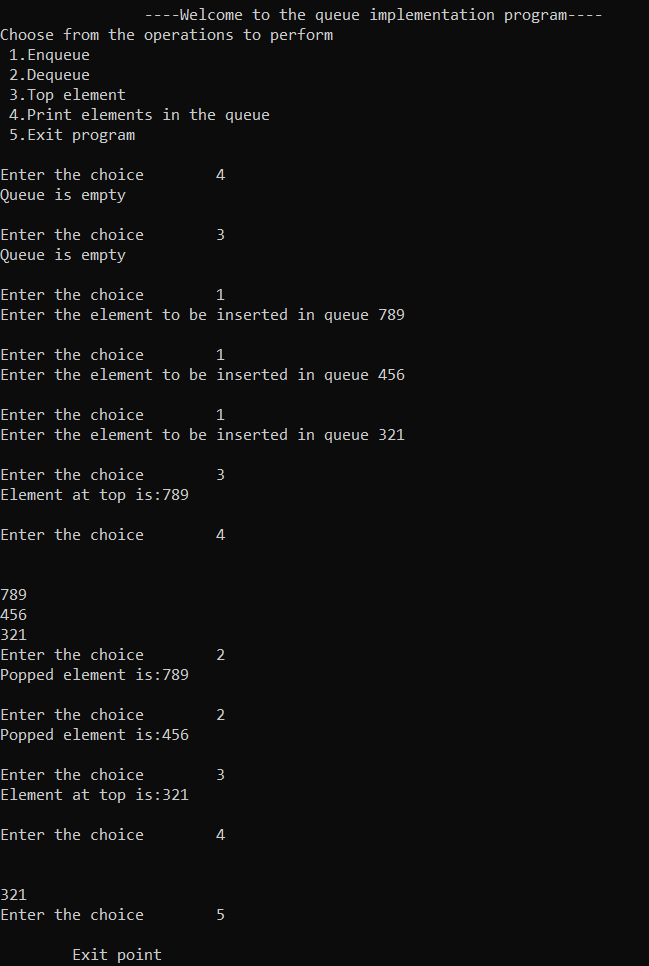
case 5: printf("\n \tExit point\n"); break;

default: {printf("Invalid choice , please choose from the available choices\n");}

} }

while(ch!=5);

return 0;}

******

**18. single linked list**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

typedefstruct node DATA\_NODE;

DATA\_NODE \*head\_node, \*first\_node, \*temp\_node = 0, \*prev\_node, next\_node;

intch ,data;

void insert() {

printf("Enter Element for Insert Linked List : \n");

scanf("%d", &data);

temp\_node = (DATA\_NODE \*) malloc(sizeof (DATA\_NODE));

temp\_node->value = data;

if (first\_node == 0) {

first\_node = temp\_node;

} else {

head\_node->next = temp\_node; }

temp\_node->next = 0;

head\_node = temp\_node;

fflush(stdin);

}

void del() {

intcountvalue, pos, i = 0;

countvalue = count();

temp\_node = first\_node;

printf("Enter Position for Delete Element : \n");

scanf("%d", &pos);

if (pos> 0 &&pos<= countvalue) {

if (pos == 1) {

temp\_node = temp\_node -> next;

first\_node = temp\_node;

printf("Deleted Successfully \n");

} else {

while (temp\_node != 0) {

if (i == (pos - 1)) {

prev\_node->next = temp\_node->next;

if(i == (countvalue - 1))

{ head\_node = prev\_node;

}

printf("Deleted Successfully \n");

break;

} else { i++;

prev\_node = temp\_node;

temp\_node = temp\_node -> next; } }

}

} else printf("Invalid Position \n");

}

void display() {

int count = 0;

temp\_node = first\_node;

printf("Elements in Linked List : \n");

while (temp\_node != 0) {

printf(" %d ", temp\_node->value);

count++;

temp\_node = temp\_node -> next;

}

printf("No Of Items In Linked List : %d\n", count);

}

int count() {

int count = 0;

temp\_node = first\_node;

while (temp\_node != 0) {

count++;

temp\_node = temp\_node -> next; }

printf("No Of Items In Linked List : %d\n", count);

return count;

}

int main()

{ printf(" \t\t----Welcome to the linked list implementation program----\n");

printf("Choose from the operations to perform\n ");

printf("1.Insert\n 2.Delete\n 3.Count elements\n 4.Print elements in the linked list\n 5.Exit program\n ");

do{printf("\nEnter the choice\t");

scanf("%d",&ch);

switch(ch){

case 1: insert(); break;

case 2: del(); break;

case 3: count(); break;

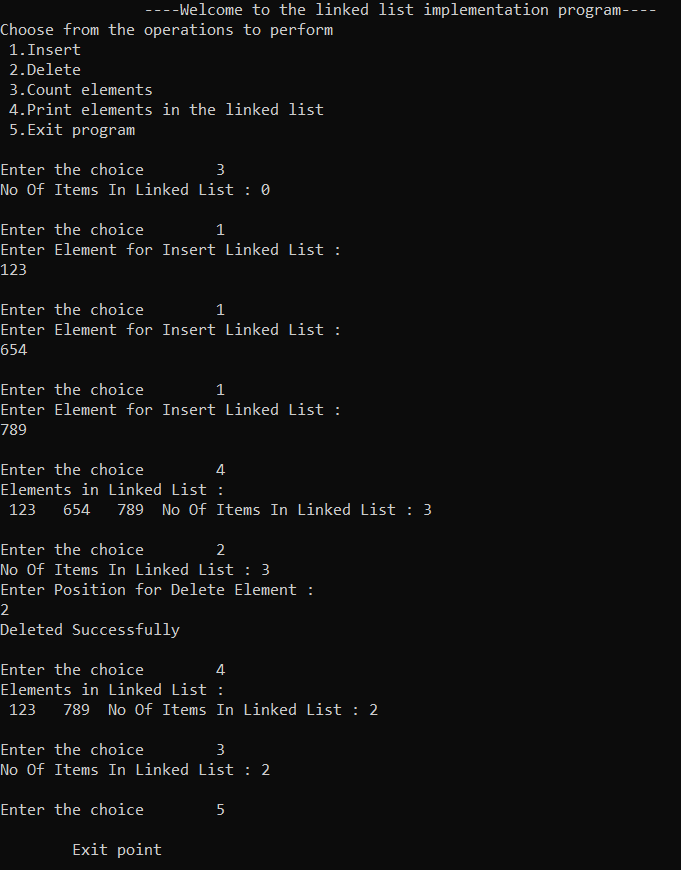
case 4: display(); break;

case 5: printf("\n \tExit point\n"); break;

default: {printf("Invalid choice , please choose from the available choices\n");} } }

while(ch!=5);

return 0;}



**19.Doubly linked list**

#include<stdio.h>

#include<stdlib.h>

struct Node;

typedefstruct Node \* PtrToNode;

typedefPtrToNode List;

typedefPtrToNode Position;

struct Node

{ int e;

Position previous;

Position next;

};

void Insert(int x, List l, Position p)

{ Position TmpCell;

TmpCell = (struct Node\*) malloc(sizeof(struct Node));

if(TmpCell == NULL)

printf("Memory out of space\n");

else

{ TmpCell->e = x;

TmpCell->previous = p;

TmpCell->next = p->next;

p->next = TmpCell; }

}

intisLast(Position p)

{ return (p->next == NULL);

}

Position Find(int x, List l)

{ Position p = l->next;

while(p != NULL && p->e != x)

p = p->next;

return p;

}

void Delete(int x, List l)

{ Position p, p1, p2;

p = Find(x, l);

if(p != NULL)

{ p1 = p -> previous;

p2 = p -> next;

p1 -> next = p -> next;

if(p2 != NULL)

p2 -> previous = p -> previous; }

else

printf("Element not found\n");

}

void Display(List l)

{ printf("The list element are : ");

Position p = l->next;

while(p != NULL)

{ printf("%d ", p->e);

p = p->next; }

}

int main()

{ int x, pos, ch, i; List l, l1;

l = (struct Node \*) malloc(sizeof(struct Node));

l->previous = NULL; l->next = NULL;

List p = l;

printf(" \t\t----Welcome to the linked list implementation program----\n");

printf("Choose from the operations to perform\n ");

printf("1.Insert\n 2.Delete\n 3.Search elements\n 4.Print elements in the linked list\n 5.Exit program\n ");

do{printf("\nEnter the choice\t");

scanf("%d",&ch);

switch(ch){

case 1: p = l;

printf("Enter the element to be inserted :: ");

scanf("%d",&x);

printf("Enter the position of the element :: ");

scanf("%d",&pos);

for(i = 1; i <pos; i++)

{ p = p->next; }

Insert(x,l,p); break;

case 2: p = l;

printf("Enter the element to be deleted :: ");

scanf("%d",&x); Delete(x,p); break;

case 3: p = l;

printf("Enter the element to be searched :: ");

scanf("%d",&x);

p = Find(x,p);

if(p == NULL)

printf("Element does not exist!!!\n");

else

printf("Element exist!!!\n");

break;

case 4: Display(l); break;

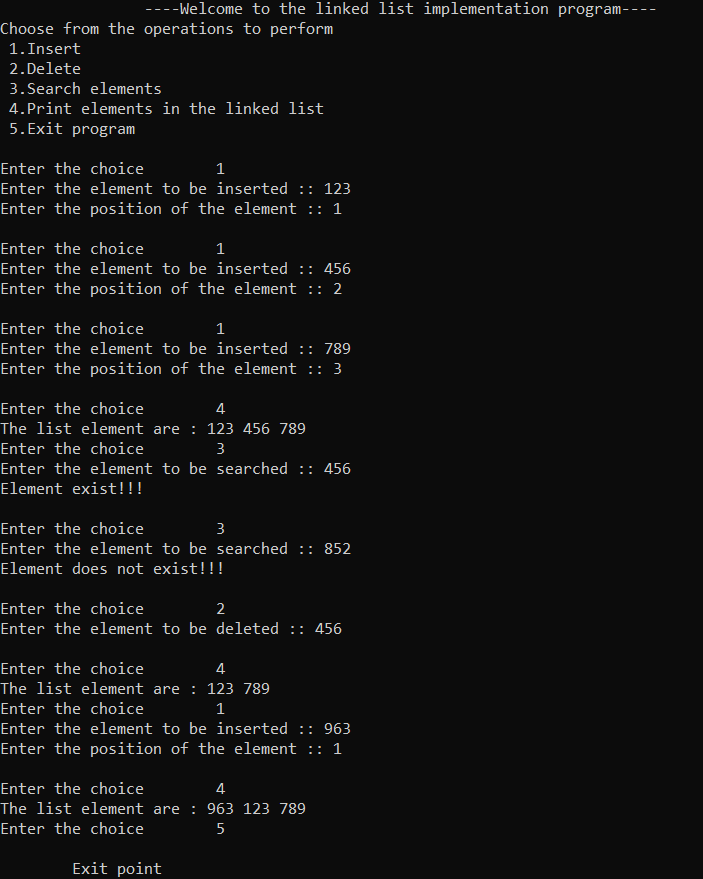
case 5: printf("\n \tExit point\n"); break;

default: {printf("Invalid choice , please choose from the available choices\n");}

} }

while(ch!=5);

return 0;}



**20.Binary search tree**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*left;

struct node \*right;

};

void check(struct node\*,struct node\*);

struct node\* newnode();

void search(struct node\*,int );

void inorder(struct node\*);

void preorder(struct node\*);

void postorder(struct node\*);

int main()

{

int c,d,trav,sear;

struct node \*top=NULL,\*temp;

do{

printf("1.insert\n2.search\n3.display\n4.quit\nenter choice=");

scanf("%d",&c);

switch(c)

{

case 1:

temp=newnode();

if (top==NULL)

top=temp;

else

check(top,temp);

break;

case 2:

printf("SEARCH ELEMENT = ");

scanf("%d",&sear);

search(top,sear);

break;

case 3:printf("1.PREORDER TRAVERSAL\n2.INORDER TRAVERSAL\n3.POST- ORDER TRAVERSAL\nENTER CHOICE = ");

scanf("%d",&trav);

if (trav==1)

preorder(top);

else if (trav==2)

inorder(top);

else if (trav==3)

postorder(top);

else

printf("INVALID CHOICE !!! \n");

printf("\n");

break;

case 4:

break;

default:

printf("enter valid!!\n");

}

}while(c!=4);

return 0;

}

struct node\* newnode()

{

int d;

struct node\* temp;

temp=(struct node\*)malloc(sizeof(struct node));

printf("ENTER ELEMENT = ");

scanf("%d",&temp->data);

temp->left=NULL;

temp->right=NULL;

return temp;

}

void check(struct node \*ptr,struct node\* temp)

{

if (temp->data<ptr->data)

{

if (ptr->left!=NULL)

check(ptr->left,temp);

else

ptr->left=temp;

}

if (temp->data>ptr->data)

{

if (ptr->right!=NULL)

check(ptr->right,temp);

else

ptr->right=temp;

}

}

void inorder(struct node\* top)

{

if (top!=NULL)

{

inorder(top->left);

printf("%d ",top->data);

inorder(top->right);

}

}

void preorder(struct node\* top)

{

if (top!=NULL)

{

printf("%d ",top->data);

preorder(top->left);

preorder(top->right);

}

}

void postorder(struct node\* top)

{

if (top!=NULL)

{

postorder(top->left);

postorder(top->right);

printf("%d ",top->data);

}

}

void search(struct node\* top,int d)

{

if (top->data==d)

printf("ELEMENT FOUND !!!\n");

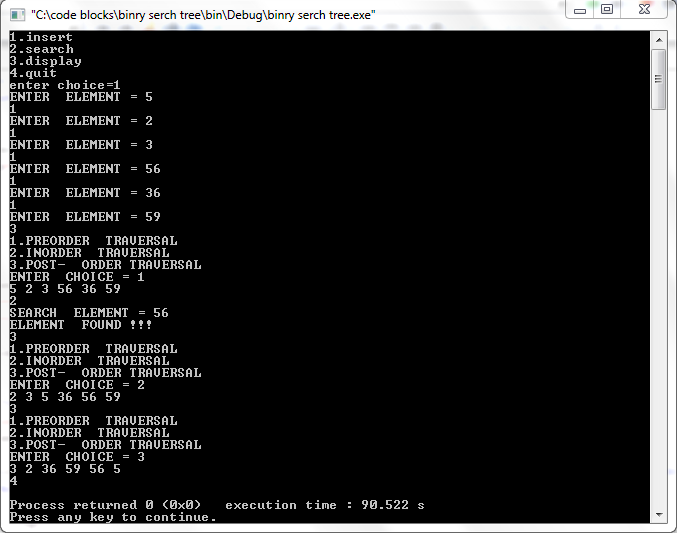
else if (top->data>d)

search(top->left,d);

else if (top->data<d)

search(top->right,d);

}



**21.Quick sort**

#include<stdio.h>

#include<stdlib.h>

int \*ar;

int sort(int,int);

int main()

{

int q,size;

printf("Enter NO. of Elements=");

scanf("%d",&size);

ar=(int\*)malloc(size\*sizeof(int));

for (q=0;q<size;q++)

{

printf("Enter Element=");

scanf("%d",&ar[q]);

}

sort(0,size-1);

for (q=0;q<size;q++)

printf("%d ",ar[q]);

return 0;

}

int sort(int start,int end)

{

int c=start-1,j,temp,w,pivot=ar[end];

if (start<end)

{

for (j=start;j<end;j++)

{

if (ar[j]<pivot)

{

c++;

temp=ar[j];

ar[j]=ar[c];

ar[c]=temp;

}

}

temp=pivot;

ar[end]=ar[c+1];

ar[c+1]=temp;

w=c+1;

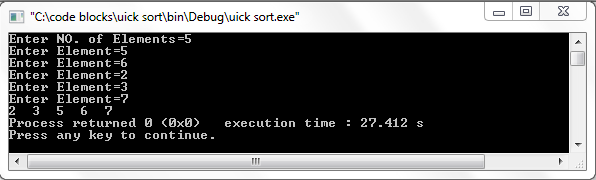
sort(start,w-1);

sort(w+1,end);

}

return 0;

}



**22.Single circular linked list**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*f = NULL;

struct node \*l= NULL;

int count=0;

void ins(int );

void search();

void display();

int main ()

{

int g,d;

do{

printf("1.INSERT\n2.DISPLAY\n3.SEARCH\n4.QUIT\nEnter your choice=");

scanf("%d",&g);

switch(g)

{

case 1:

printf("Enter=");

scanf("%d",&d);

ins(d);

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

break;

default:

printf("Invalid Input\n");

}

}while(g!=4);

return 0;

}

void ins(int x)

{

struct node \*t;

t=(struct node \*)malloc(sizeof(struct node));

if (l==NULL && f==NULL)

{

l=t;

f=t;

l->data=x;

l->next=f;

}

else

{

t->data=x;

l->next=t;

t->next=f;

l=t;

}

}

void search()

{

struct node \*t;

int d;

t=f;

printf("SEARCH ELEMENT --->");

scanf("%d",&d);

do

{

if (t->data==d)

{

printf("SEARCH SUCCESSFUL !!\n");

return ;}

t=t->next;

}

while (t!=f);

printf("SEARCH UNSUCCESSFUL !!!\n");

printf("\n");

}

void display()

{

struct node \*t;

t=f;

do

{

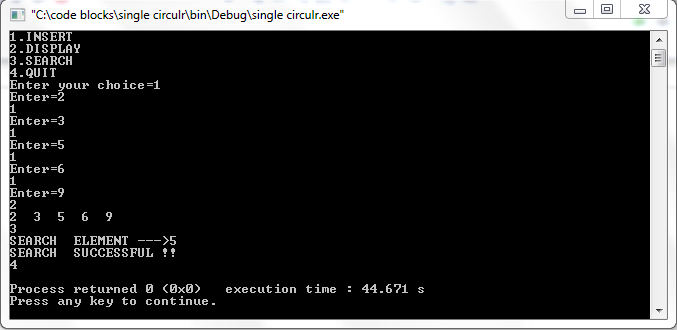
printf("%d ",t->data);

t=t->next;

}

while (t!=f);

printf("\n");

} 

**23.Double circular linked list**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*f = NULL;

struct node \*l= NULL;

int count=0;

void ins(int );

void search();

void display();

int main ()

{

int g,d;

do{

printf("1.INSERT\n2.DISPLAY\n3.SEARCH\n4.QUIT\nEnter your choice=");

scanf("%d",&g);

switch(g)

{

case 1:

printf("Enter=");

scanf("%d",&d);

ins(d);

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

break;

default:

printf("Invalid Input\n");

}

}while(g!=4);

return 0;

}

void ins(int x)

{

struct node \*t;

t=(struct node \*)malloc(sizeof(struct node));

if (l==NULL && f==NULL)

{

l=t;

f=t;

l->data=x;

l->next=f;

}

else

{

t->data=x;

l->next=t;

t->next=f;

l=t;

}

}

void search()

{

struct node \*t;

int d;

t=f;

printf("SEARCH ELEMENT --->");

scanf("%d",&d);

do

{

if (t->data==d)

{

printf("SEARCH SUCCESSFUL !!\n");

return ;}

t=t->next;

}

while (t!=f);

printf("SEARCH UNSUCCESSFUL !!!\n");

printf("\n");

}

void display()

{

struct node \*t;

t=f;

do

{

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

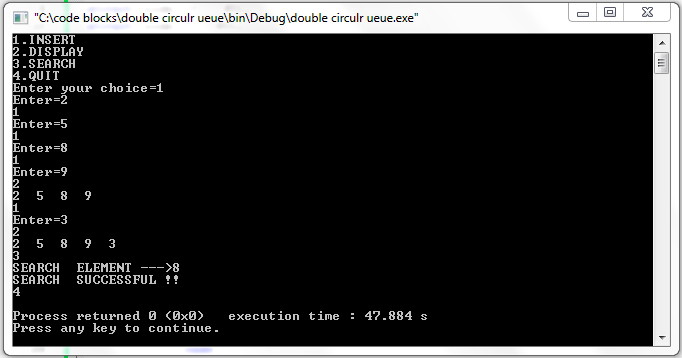
t=t->next;

}

while (t!=f);

printf("\n");

}



**24.MIN HEAP**

#include<stdio.h>

#include<stdlib.h>

void insert();

void reheapup(int,int);

void display();

int \*ar,j=0,m=0;

int main()

{

int c,x=0,a;

printf("Enter Total NO. of elements=");

scanf("%d",&a);

ar=(int\*)malloc(a\*sizeof(int));

do

{

printf("1.INSERT\n2.DISPLAY\n3.QUIT\nENTER choice=");

scanf("%d",&c);

switch(c)

{

case 1:

if (x<a)

{

insert();

x++;

}

else

printf("HEAP IS FULL !! \n");

break;

case 2:

display();

break;

case 3:

break;

default:

printf("INValid input!!\n");

}

}while (c!=3);

return 0;

}

void insert()

{

int d;

printf("Enter the Element=");

scanf("%d",&d);

ar[j]=d;

if (j!=0)

reheapup(d,m);

m++;

j++;

}

void reheapup(int d,int s)

{

int temp;

if (ar[(s-1)/2]>d)

{

temp=ar[(s-1)/2];

ar[(s-1)/2]=d;

ar[s]=temp;

s=(s-1)/2;

reheapup(d,s);

}

else

return;

}

void display()

{

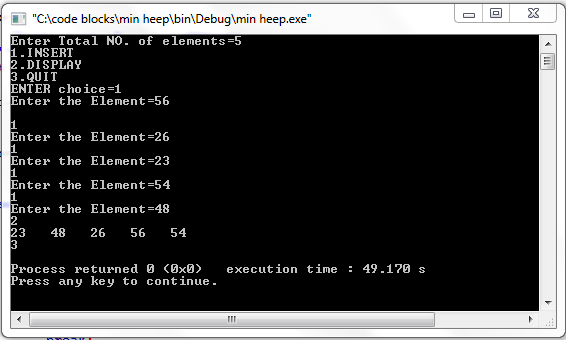
int x;

for (x=0;x<j;x++)

printf("%d ",ar[x]);

printf("\n");

}



**25.MAX HEAP**

#include<stdio.h>

#include<stdlib.h>

void insert();

void reheapup(int,int);

void display();

void delete();

int \*ar,j=0,m=0;

int main()

{

int c,x=0,a;

printf("Enter Total NO. of elements=");

scanf("%d",&a);

ar=(int\*)malloc(a\*sizeof(int));

do

{

printf("1.INSERT\n2.DISPLAY\n3.DELETE\n4.QUIT\nENTER choice=");

scanf("%d",&c);

switch(c)

{

case 1:

if (x<a)

{

insert();

x++;

}

else

printf("HEAP IS FULL!!! \n");

break;

case 2:

display();

printf("\n%d",j);

break;

case 3:

delete();

break;

case 4:

break;

default:

printf("INValid input!!\n");

}

}while (c!=4);

return 0;

}

void insert()

{

int d;

printf("Enter the Element=");

scanf("%d",&d);

ar[j]=d;

if (j!=0)

reheapup(d,m);

m++;

j++;

}

void reheapup(int d,int s)

{

int temp;

if (ar[(s-1)/2]<d)

{

temp=ar[(s-1)/2];

ar[(s-1)/2]=d;

ar[s]=temp;

s=(s-1)/2;

reheapup(d,s);

}

else

return;

}

void delete()

{

int n=j-1,last,left,right,loc=0;

last=ar[n];

n=n-1;

left=2\*loc+1;

right=2\*loc+2;

while (right<=n)

{

if (last>=ar[left] && last>=ar[right])

{

ar[loc]=last;

j--;

return;

}

if (ar[right]<=ar[left])

{

ar[loc]=ar[left];

loc=left;

}

else

{

ar[loc]=ar[right];

loc=right;

}

left=2\*loc+1;

right=2\*loc+2;

}

if (left==n && ar[left]>last)

{

ar[loc]=ar[left];

loc=left;

}

ar[loc]=last;

j--;

}

void display()

{

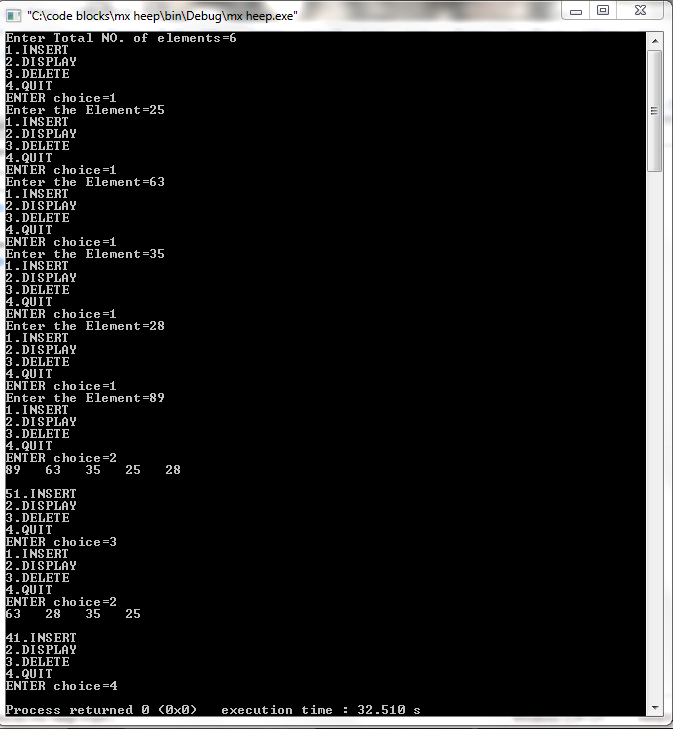
int x;

for (x=0;x<j;x++)

printf("%d ",ar[x]);

printf("\n");

}



**26.HEAP SORT**

#include<stdio.h>

#define MAX 50

void insert();

void reheapup(int,int);

void display();

void sort();

int \*ar,j=0,m=0;

int main()

{

int c,x,p;

printf("Enter Total NO. of elements=");

scanf("%d",&x);

ar=(int\*)malloc(x\*sizeof(int));

for (p=0;p<x;p++)

insert();

sort();

printf("SORTED HEAP ------->>>");

display();

return 0;

}

void insert()

{

int d;

printf("Enter the Element=");

scanf("%d",&d);

ar[j]=d;

if (j!=0)

reheapup(d,m);

m++;

j++;

}

void reheapup(int d,int s)

{

int temp;

if (ar[(s-1)/2]<d)

{

temp=ar[(s-1)/2];

ar[(s-1)/2]=d;

ar[s]=temp;

s=(s-1)/2;

reheapup(d,s);

}

else

return;

}

void display()

{

int x;

for (x=0;x<j;x++)

printf("%d ",ar[x]);

printf("\n");

}

void sort()

{

int u,h,temp,b;

for (h=j-1;h>0;h--)

{

temp=ar[0];

ar[0]=ar[h];

ar[h]=temp;

for (u=h-1;u>=0;u--)

{

reheapup(ar[u],u);

}

}

}

