**/\*1. Program to sorted a list of n-integer using the Quick sort\*/**

#include<stdio.h>

#include<conio.h>

void quicksort(int [],int,int);

int partition(int [],int,int);

void main()

{

int a[20],n,i;

clock\_t start,end;

double cpu\_time\_used;

clrscr();

printf("\n enter the number of elements\n");

scanf("%d",&n);

printf("\n enter the %d elements \n",n);

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

scanf("%d",&a[i]);

quicksort(a,0,n-1);

printf("\n sorted array is");

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

printf("\n %d",a[i]);

printf("\n");

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f second",cpu\_time\_used);

getch();

}

void quicksort(int a[],int low,int high)

{

int j;

if(low<high)

{

j=partition(a,low,high);

quicksort(a,low,j-1);

quicksort(a,j+1,high);

}

}

int partition(int a[],int low,int high)

{

int i,j,temp,key;

key=a[low];

i=low+1;

j=high;

while(1)

{

while(i<high && key>=a[i])

i++;

while(key<a[j])

j--;

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

else

{

temp=a[low];

a[low]=a[j];

a[j]=temp;

return j;

}

}

}

**/\*2. Program to sorted a list of n-integer using the heap sort\*/**

#include<stdio.h>

#include<conio.h>

#include<time.h>

createheap(int[],int);

heapsort(int[],int);

void main()

{

int k[10],i,n;

clock\_t start,end'

double cpu\_time\_used;

clrscr();

printf("enter how many elements\n");

scanf("%d",&n);

printf("enter array elements\n");

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

scanf("%d",&k[i]);

printf("\n original list\n");

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

printf("%4d",k[i]);

heapsort(k,n);

printf("\n sorted list\n");

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

printf("%4d",k[i]);

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f",cpu\_time\_used);

return;

}

createheap(int k[],int n)

{

int temp,q,i,j,key;

for(q=2;q<=n;q++)

{

i=q;

key=k[q];

j=i/2;

while((i>1)&&(key>k[j]))

{

temp=k[j];

k[j]=k[i];

k[i]=temp;

i=j;

j=i/2;

if(j<1)

j=1;

}

k[i]=key;

}

return(0);

}

heapsort(int k[],int n)

{

int temp,q,i,j,key;

createheap(k,n);

for(q=n;q>=2;q--)

{

temp=k[q];

k[q]=k[1];

k[1]=temp;

i=1;

j=2;

key=k[1];

if(j+1<q)

{

if(k[j+1]>k[j])

j++;

}

while((j<=(q-1))&&(k[j]>key))

{

temp=k[j];

k[j]=k[i];

k[i]=temp;

i=j;

j=2\*i;

if(j+1<q)

{

if(k[j+1]>k[j])

j++;

else

if(j>n)

j=n;

}

k[i]=key;

}

}

return(0);

}

**/\*3.program to sort a list of n-integer using merge sort\*/**

#include<stdio.h>

#include<conio.h>

#include<stime.h>

void sort(int[],int);

void main()

{

int a[10],b[10],c[25],m,n,i,j,k,temp;

clock\_t start,end'

double cpu\_time\_used;

clrscr();

start=clock();

printf("enter the size of a\n");

scanf("%d",&m);

printf("enter the elements into array a\n");

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

scanf("%d",&a[i]);

sort(a,m);

printf("the sorted array is:\n");

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

printf("%d\n",a[i]);

printf("enter the size of b\n");

scanf("%d",&n);

printf("enter the elements into array b\n");

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

scanf("%d",&b[i]);

sort(b,n);

printf("the sorted array is\n");

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

printf("%d\n",b[i]);

i=1,j=1,k=1;

while(i<m &&j<=n)

{

if(a[i]<b[j])

{

c[k]=a[i];

i++,k++;

}

else

{

c[k]=b[j];

j++,k++;

}

}

while(i<=m)

{

c[k]=a[i];

i++,k++;

}

while(j<=n)

{

c[k]=b[j];

j++,k++;

}

printf("elements after merging are\n");

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

printf("%d\n",c[i]);

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f",cpu\_time\_used);

getch();

}

void sort(int a[10],int m)

{

int i,j,temp;

for(i=1;i<=m-1;i++)

for(j=1;j<=m-i;j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

**/\*4. C program to print all nodes reachable from given source in a digraph using BFS\*/**

#include<stdio.h>

#include<conio.h>

#include<time.h>

void read();

void print();

void bfs\_traversal();

int n,s[20],a[20][20],source;

void main()

{

clock\_t start,end;

double cpu\_time\_used;

clrscr();

start=clock();

read();

bfs\_traversal();

print();

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f second",cpu\_time\_used);

getch();

}

void read()

{

int i,j;

printf("\nenter the number of nodes\n");

scanf("%d",&n);

printf("enter adjacency matrix");

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

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

scanf("%d",&a[i][j]);

printf("\nenter the source\n");

scanf("%d",&source);

}

void bfs\_traversal()

{

int f,r,q[20],u,v,i;

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

s[i]=0;

f=r=0;

q[r]=source;

s[source]=1;

while(f<=r)

{

u=q[f++];

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

{

if(a[u][v]==1 && s[v]==0)

{

s[v]=1;

q[++r]=v;

}

}

}

}

void print()

{

int i;

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

{

if(s[i]==0)

printf("vertex %d is not reachable", i);

else

printf("vertex %d is reachable",i);

}

}

**/\*5. Program to implement preorder, postorder and inorder traversal of a binary search tree using linked list\*/**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<time.h>

struct Tree

{

int info;

struct Tree \*left;

struct Tree \*right;

};

typedef struct Tree \*TNODE;

TNODE CreateTree(TNODE, int);

void preorder(TNODE);

void inorder(TNODE);

void postorder(TNODE);

void main()

{

char e;

int choice, done;

TNODE p;

clock\_t start,end;

double cpu\_time\_used;

clrscr();

p= NULL; done = 0;

while(!done)

{

printf("\n 1. creat \n 2. Preorder\n 3. Inorder\n 4. Postorder\n 5. exit\n");

printf("enter choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("\n enter the element\n");

scanf("%d",&e);

p =CreateTree(p,e);

break;

case 2: preorder(p);

break;

case 3: inorder (p);

break;

case 4: postorder(p);

break;

case 5: done=1;

break;

default:

printf("\n invalid choice\n");

break;

}

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f",cpu\_time\_used);

}

}

TNODE CreateTree(TNODE p, int e)

{

if(p==NULL)

{

p=(TNODE)malloc(sizeof(struct Tree));

p->info=e;

p->left=p->right=NULL;

}

else

if(e==p->info)

printf("\n duplicate entry\n");

else

if(e<p->info)

p->left=CreateTree(p->left,e);

else

p->right=CreateTree(p->right,e);

return p;

}

void preorder(TNODE p)

{

if(p!=NULL)

{

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

preorder(p->left);

preorder(p->right);

}

}

void inorder(TNODE p)

{

if(p!=NULL)

{

inorder(p->left);

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

inorder(p->right);

}

}

void postorder(TNODE p)

{

if(p!=NULL)

{

postorder(p->left);

postorder(p->right);

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

}

}

**/\*6. Write a program to illustrate singly linked list\*/**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<time.h>

struct NODE

{

int info;

struct NODE \*next;

};

void insertf(struct NODE\*\*,int);

int deletef(struct NODE \*\*);

void disp(struct NODE \*);

void main()

{

struct NODE \*first=NULL;

int x,opt=0;

clock\_t start,end;

double cpu\_time\_used;

clrscr();

start=clock();

while(opt<4)

{

printf("stack using linked list \n");

printf("1.push \n2.pop \n3.display \n4.exit \n");

printf("enter option \n");

scanf("%d",&opt);

switch(opt)

{

case 1:printf("\n enter item \n");

scanf("%d",&x);

insertf(&first,x);

break;

case 2:x=deletef(&first);

if(x!='\0')

{

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

getch();

}

case 3:printf("content of stack \n");

disp(first);

}

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f",cpu\_time\_used);

}

}

void insertf(struct NODE \*\*first,int x)

{

struct NODE \*newl;

newl=(struct NODE \*)malloc(sizeof(struct NODE));

newl->info=x;

newl->next=\*first;

\*first=newl;

}

int deletef(struct NODE \*\*first)

{

struct NODE \*temp;

int x;

if(\*first==NULL)

{

if("stack underflow");

return('\0');

}

temp=\*first;

\*first=temp->next;

x=temp->info;

free(temp);

return x;

}

void disp(struct NODE \*first)

{

if(first==NULL)

printf("empty stack \n");

else

while(first!=NULL)

{

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

first=first->next;

}

getch();

}

**/\*7. Write a program to illustrate circular linked list\*/**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define NULL 0

#include<time.h>

struct node

{

int data;

struct node \*link;

};

struct node \*head, \*p, \*front, \*rear, \*last;

main()

{

void insert(int);

void display();

int del();

int item,info,ch;

clock\_t start,end;

double cpu\_time\_used;

clrscr();

start clock();

front=rear=NULL;

while(1)

{

printf("\n circular linked list menu");

printf("\n 1. insert");

printf("\n 2. delete") ;

printf("\n 3. display");

printf("\n 4. exit");

printf("\n enter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("\n enter the element to be added to the list");

scanf("%d",&info);

insert(info);

break;

case 2: item=del();

printf("\n deleted item is %d",item);

break;

case 3:printf("\n elements in the circular linked list");

display();

break;

case 4: return(0);

default:printf("\n error in choice");

break;

}

end=clock();

cpu\_time\_used=((double)(end-start));

printf("\n cpu\_time\_used %f",cpu\_time\_used);

}

}

void insert(int num)

{

(void \*)p = malloc(sizeof(node));

p->data = num;

p->link = NULL;

if(front==NULL)

front = p;

else

rear ->link = p;

rear = p;

rear->link = front;

}

void display()

{

p = front;

if(p==NULL)

{

printf("\n list is empty");

return;

}

while(p!=rear)

{

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

p = p->link;

}

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

}

int del()

{

int item=0;

if(front==NULL)

{

printf("\n list is empty");

getch();

}

else

{

if(front == rear)

{

item = front ->data;

free(front);

front = NULL;

rear = NULL;

}

else

{

p = front;

item = p->data;

front = front->link;

rear ->link = front;

free(p);

}

}

return(item);

}