Exercise 1:

/\* write recursive c programme which computes the nth fibanocci number,for appropriate values of n. \*/

#include<stdio.h>

#include<conio.h>

int fibonnaci(int n);

main()

{

int i,s,x;

clrscr();

printf("enter the size of the fibonnaci series \n");

scanf("%d",&s);

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

{

x=fibonnaci(i);

printf("%4d",x);

}

getch();

}

int fibonnaci(int n)

{

if(n==0||n==1)

return n;

else

return fibonnaci(n-1)+fibonnaci(n-2);

}

Exercise 2:

a)write recursive c programme for calculation of factorial of an integer.

#include<stdio.h>

#include<conio.h>

int factorial(int n);

main()

{

int factorial,n;

clrscr();

printf("enter n value");

scanf("%d",&n);

factorial=fact(n);

printf("factorial=%d \n",factorial);

getch();

}

fact(int n)

{

if(n==0)

return 1;

else

return n\*fact(n-1);

}

b)write recursive c programme for calculation of GCD(n,m)

#include<stdio.h>

#include<conio.h>

int gcd(int a,int b);

main()

{

int gcdresult,a,b;

clrscr();

printf("enter aand b values");

scanf("%d%d",&a,&b);

printf("gcd of a,b before test \n");

gcdresult=gcd(a,b);

printf("the gcdresult is %d \n",gcdresult);

printf("the gcdresult is ending \n");

getch();

}

int gcd(int a,int b)

{

if(b==0)

return a;

else if(a==0)

return b;

else

return gcd(b,a%b);

}

c)write recursive c programme for towers of hanoi.

#include<stdio.h>

#include<conio.h>

towers(int numdisks,char source,char dest,char auxilary);

int count=0;

main()

{

int n;

clrscr();

printf("enter number of disks \n");

scanf("%d",&n);

printf("towers of hanoi is starting \n");

towers(n,'a','b','c');

getch();

}

towers(int numdisks,char source,char dest,char auxilary)

{

printf("towers(%d,%c,%c,%c) \n",numdisks,source,dest,auxilary);

if(numdisks==1)

printf("\t\t\t step %d:move from %c to %c \n",++count,source,dest);

else

{

towers(numdisks-1,source,auxilary,dest);

printf("\t\t\t step %d:move from %c to %c \n",++count,source,dest);

towers(numdisks-1,auxilary,dest,source);

}

}

Exercise 3:

a)rite a C program that use recursive functions to perform linear search for a keyvalue in a given list.

#include<stdio.h>

#include<conio.h>

int reclinearsearch(int a[],int length,int key);

main()

{

int a[50],n,x,i,element;

clrscr();

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

scanf("%d",&n);

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

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

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

printf("enter the element for searching \n");

scanf("%d",&x);

element=reclinearsearch(a,n,x);

if(element==-1)

printf("element not found \n");

else

printf("element found at %d \n",element);

getch();

}

int reclinearsearch(int a[],int length,int key)

{

if(length==0)

return -1;

else if(a[length-1]==key)

return length;

else

return reclinearsearch(a,length-1,key);

}

b)write a C program that use non recursive functions to perform linear search for a keyvalue in a given list.

#include<stdio.h>

#include<conio.h>

int nonreclinearsearch(int a[],int n,int key);

main()

{

int a[50],size,x,p,i;

clrscr();

printf("enter the size of array");

scanf("%d",&size);

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

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

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

printf("enter the element for searching \n");

scanf("%d",&x);

p=nonreclinearsearch(a,size,x);

if(p!=1)

printf("%d element found at %d \n",x,p);

else

printf("%d element is not found \n",x);

getch();

}

int nonreclinearsearch(int a[],int n,int key)

{

int i;

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

{

if(a[i]==key)

return i+1;

}

return -1;

}

c)write a C program that use recursive functions to perform binary search for a keyvalue in a given list.

#include<stdio.h>

#include<conio.h>

int recbinsearch(int a[],int low,int high,int key);

main()

{

int a[50],size,x,i,p;

clrscr();

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

scanf("%d",&size);

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

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

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

printf("enter the element for searching \n");

scanf("%d",&x);

p=recbinsearch(a,0,size-1,x);

if(p!=-1)

printf("element found at %d \n",p);

else

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

getch();

}

int recbinsearch(int a[],int low,int high,int key)

{

int mid;

if(low>high)

return -1;

mid=(low+high)/2;

if(a[mid]==key)

return mid;

else if(key<a[mid])

return recbinsearch(a,low,mid-1,key);

else

return recbinsearch(a,mid+1,high,key);

}

d)write a C program that use non recursive functions to perform binary search for a keyvalue in a given list

#include<stdio.h>

#include<conio.h>

int nonrecbinsearch(int a[],int n,int key);

main()

{

int a[50],size,x,p,i;

clrscr();

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

scanf("%d",&size);

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

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

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

printf("enter the element for searching \n");

scanf("%d",&x);

p=nonrecbinsearch(a,size,x);

if(p!=1)

printf("%d element found at %d \n",x,p);

else

printf("%d element is not found \n",x);

getch();

}

int nonrecbinsearch(int a[],int n,int key)

{

int low,high,mid;

low=0;

high=n-1;

while(low<=high)

{

mid=(low+high)/2;

if(a[mid]==key)

return mid;

else if(key<a[mid])

high=mid-1;

else

low=mid+1;

}

return -1;

})

Exercise 4:

a)write a C program that implement bubble sort,to sort a given list of integers in ascending order.

#include<stdio.h>

#include<conio.h>

int bubblesort(int a[],int n);

main()

{

int size,i,a[50];

clrscr();

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

scanf("%d",&size);

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

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

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

bubblesort(a,size);

getch();

}

int bubblesort(int a[],int n)

{

int i,j,temp;

for(i=1;i<=n-1;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;

}

}

b)write a C program that implement insertion sort,to sort a given list of integers in ascending order.#include<stdio.h>

#include<conio.h>

int insertionsort(int a[],int n);

main()

{

int size,i,a[50];

clrscr();

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

scanf("%d",&size);

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

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

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

insertionsort(a,size);

getch();

}

int insertionsort(int a[],int n)

{

int i,pos,key;

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

{

key=a[i];

pos=i;

while(pos>1&&a[pos-1]>key)

{

a[pos]=a[pos-1];

pos=pos-1;

a[pos]=key;

}

}

printf("the sorted elements are \n");

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

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

}

c)write a C program that implement quick sort,to sort a given list of integers in ascending order.

#include<stdio.h>

#include<conio.h>

int quicksort(int a[],int first,int last);

int partition(int a[],int first,int last);

main()

{

int n,i,a[50];

clrscr();

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

scanf("%d",&n);

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

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

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

quicksort(a,1,n);

printf("elements after sorting \n");

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

printf("%3d",a[i]);

getch();

}

int quicksort(int a[],int first,int last)

{

int loc;

if(first<last)

{

loc=partition(a,first,last);

quicksort(a,first,loc-1);

quicksort(a,loc+1,last);

}

}

int partition(int a[],int first,int last)

{

int left,right,pivot,temp;

left=first;

right=last+1;

pivot=a[first];

while(left<right)

{

do

left=left+1;

while(a[left]<=pivot);

do

right=right-1;

while(a[right]>pivot);

if(left<right)

{

temp=a[left];

a[left]=a[right];

a[right]=temp;

}

}

temp=a[first];

a[first]=a[right];

a[right]=temp;

return right;

}

}

printf("the sorted elements are \n");

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

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

}

Exercise 5:

a)write a C program that implement heap sort,to sort a given list of integers in descending order.

#include<stdio.h>

#include<conio.h>

void maxheap();

void heapsort();

void swap(int \*p,int \*q);

int a[30],i,j,k,n;

main()

{

clrscr();

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

scanf("%d",&n);

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

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

{

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

maxheap();

}

printf("maxheap elements are \n");

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

printf("%3d",a[i]);

printf("elements after sorting \n");

heapsort();

getch();

}

void swap(int \*p,int \*q)

{

int t;

t=\*p;

\*p=\*q;

\*q=t;

}

void maxheap()

{

for(j=0;j<=i/2;j++)

if(a[j]<a[2\*j+1]||a[j]<a[2\*j+2])

{

if(a[2\*j+1]<a[2\*j+2])

swap(&a[j],&a[2\*j+2]);

else

swap(&a[j],&a[2\*j+1]);

maxheap();

}

}

void heapsort()

{

k=0;

if(k<n)

{

printf("%3d",a[0]);

swap(&a[0],&a[n-1]);

a[n-1]=0;

n--;

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

maxheap();

heapsort();

}

}

b)write a C program that implement radix sort,to sort a given list of integers in ascending order

#include<stdio.h>

#include<stdlib.h>

void radix(int a[],int n,int m)

{

typedef struct rad

{

int data;

struct rad \* next;

}node;

node \* ptr,\*start,\*prev;

node \*front[10], \*rear[10];

int k=1,i,j,y,p=0;

/\*creating initial linked list\*/

start=NULL;

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

{

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

ptr->data=a[i];

ptr->next=NULL;

if(start==NULL)

prev=start=ptr;

else

prev->next=ptr;

prev=ptr;

}

/\*radix sort\*/

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

{

for(j=0;j< 10;++j)

front[j]=NULL;

/\*placing elements into queues\*/

ptr=start;

while(ptr!=NULL)

{

y=ptr->data/k %10;/\*y is the digit\*/

if(front[y]==NULL)

{

front[y]=ptr;

rear[y]=ptr;

}

else

{

rear[y]->next=ptr;

rear[y]=ptr;

}

ptr=ptr->next;

}

start=NULL;

for(j=0;j< 10;++j)

if(front[j]!=NULL)

{

if(start==NULL)

start=front[j];

else rear[p]->next=front[j];

p=j;

}

rear[p]->next=NULL;

k=k\*10;

}

/\*copying back to array\*/

ptr=start;

for(i=0;i< n;++i,ptr=ptr->next)

a[i]=ptr->data;

}

void main()

{

int a[100],n,i,m;

char temp;

clrscr();

printf("ENTER NUMBER OF NUMBERS AND NUMBER OF DIGITS\n");

scanf("%d%d",&n,&m);

printf("ENTER ELEMENTS\n");

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

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

radix(a,n,m);

printf("SORTED LIST\n");

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

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

getch();

}

c)write a C program that implement merge sort,to sort a given list of integers in ascending order.

#include<stdio.h>

#include<conio.h>

int mergesort(int a[],int first,int last);

int merge(int a[],int first,int mid,int last);

main()

{

int size,i,a[50];

clrscr();

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

scanf("%d",&size);

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

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

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

mergesort(a,1,size);

printf("elements after sorting \n");

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

printf("%3d",a[i]);

getch();

}

int mergesort(int a[],int first,int last)

{

int mid;

if(first<last)

{

mid=(first+last)/2;

mergesort(a,first,mid);

mergesort(a,mid+1,last);

merge(a,first,mid,last);

}

}

int merge(int a[],int first,int mid,int last)

{

int first1,last1,first2,last2,k,j,temp[30];

first1=first;

last1=mid;

first2=mid+1;

last2=last;

k=first;

while(first1<=last1&&first2<=last2)

{

if(a[first1]<=a[first2])

{

temp[k]=a[first1];

k=k+1;

first1=first1+1;

}

else

{

temp[k]=a[first2];

k=k+1;

first2=first2+1;

}

}

while(first1<=last1)

{

temp[k]=a[first1];

k=k+1;

first1=first1+1;

}

while(first2<=last2)

{

temp[k]=a[first2];

k=k+1;

first2=first2+1;

}

for(j=first;j<=last;j++)

{

a[j]=temp[j];

}

}

Exercise 6:

a)write a C program that implement stack (its operations) using arrays.

#include<stdio.h>

#include<conio.h>

void push(int item);

void pop();

void display();

int stack[30],top=0,n,x;

main()

{

int ch;

clrscr();

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

scanf("%d",&n);

while(1)

{

printf("\n 1.push \n");

printf("2.pop \n");

printf("3.display \n");

printf("4.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("enter the elements to push \n");

scanf("%d",&x);

push(x);

break;

case 2:pop();

break;

case 3:display();

break;

default :exit(0);

}

}

}

void push(int item)

{

if(top==n)

printf("stack is overflow");

else

{

top=top+1;

stack[top]=item;

}

}

void pop()

{

if(top==0)

printf("stack is underflow \n");

else

{

x=stack[top];

top=top-1;

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

}

}

void display()

{

int i;

if(top==0)

printf("stack is empty \n");

else

{

for(i=top;i>0;i--)

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

}

}

b)write a C program that implement stack (its operations) using linked list.

#include<stdio.h>

#include<conio.h>

void push(void);

void pop();

void display();

int n,x;

typedef struct sll

{

int data;

struct sll \*link;

}node;

node \*curr,\*top,\*newnode;

main()

{

int ch;

top=NULL;

clrscr();

while(1)

{

printf("\n1.push \n");

printf("2.pop \n");

printf("3.display \n");

printf("4.exit\n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

default :

exit(0);

}

}

}

void push(void)

{

int x;

printf("enter the elements of node to push \n");

scanf("%d",&x);

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

newnode->data=x;

newnode->link=NULL;

if(top==NULL)

top=newnode;

else

{

newnode->link=top;

top=newnode;

}

}

void pop()

{

if(top==NULL)

printf("stack is underflow \n");

else

{

curr=top;

top=top->link;

printf("the poped element is %d \n",curr->data);

free(curr);

}

}

void display()

{

if(top==NULL)

printf("stack is underflow \n");

curr=top;

while(curr!=NULL)

{

printf("%3d",curr->data);

printf("->");

curr=curr->link;

}

printf("null");

}

Exercise 7:

a)write a C program that uses stack operations to convert infix expression into postfix expression.

#include<stdio.h>

#include<conio.h>

char stack[20],exp[30],top=-1;

void push(int);

void pop();

int isp();

int icp();

int i;

main()

{

clrscr();

printf("enter the infix expressions \n");

gets(exp);

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

{

if((exp[i]>='a'&&exp[i]<='z')||(exp[i]>='A'&&exp[i]<='Z'))

putchar(exp[i]);

else

if(top==-1)

push(exp[i]);

else

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

{

do

pop();

while(stack[top]!='(');

top=top-1;

}

else

while(1)

{

if(isp()>=icp())

pop();

else

{

push(exp[i]);

break;

}

if(top==-1)

{

push(exp[i]);

break;

}

}

}

while(top!=-1)

pop();

getch();

}

void push(int x)

{

top=top+1;

stack[top]=x;

}

void pop()

{

putchar(stack[top]);

top=top-1;

}

int isp()

{

int y;

if(stack[top]=='-'||stack[top]=='+')

y=1;

if(stack[top]=='/'||stack[top]=='\*')

y=2;

if(stack[top]=='(')

y=0;

return y;

}

int icp()

{

int y;

if(exp[i]=='+'||exp[i]=='-')

y=1;

if(exp[i]=='\*'||exp[i]=='/')

y=2;

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

y=3;

return y;

}

b)write a C program that implement queue (its operations) using arrays.

#include<stdio.h>

#include<conio.h>

void insert(int item);

void delete();

void display();

int q[30],n,x,front=0,rear=0,item,i;

main()

{

int ch;

clrscr();

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

scanf("%d",&n);

while(1)

{

printf("\n1.insert \n");

printf("2.delete\n");

printf("3.display \n");

printf("4.exit\n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("enter the elements to insert \n");

scanf("%d",&x);

insert(x);

break;

case 2:

delete();

break;

case 3:

display();

break;

default :

exit();

}

}

}

void insert(int item)

{

if(rear==n)

printf("queue is overflow");

else

{

rear=rear+1;

q[rear]=item;

}

}

void delete()

{

if(front==rear)

printf("queue is empty \n");

else

{

front=front+1;

item=q[front];

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

}

}

void display()

{

if(front==rear)

printf("queue is empty \n");

else

{

for(i=front+1;i<=rear;i++)

printf("%3d",q[i]);

}

}

c)write a C program that implement queue (its operations) using linked list.

#include<stdio.h>

#include<conio.h>

void insert(void);

void delete();

void display();

typedef struct sll

{

int data;

struct sll \*link;

}node;

node \*curr,\*front,\*rear,\*newnode;

main()

{

int ch;

clrscr();

front=NULL;

rear=NULL;

while(1)

{

printf("\n1.insert \n");

printf("2.delete\n");

printf("3.display \n");

printf("4.exit\n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

default :

exit(0);

}

}

}

void insert(void)

{

int x;

printf("enter the elements of node to push \n");

scanf("%d",&x);

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

newnode->data=x;

newnode->link=NULL;

if(rear==NULL)

front=rear=newnode;

else

{

rear->link=newnode;

rear=newnode;

}

}

void delete()

{

if(front==NULL)

printf("queue is empty \n");

else

{

curr=front;

front=front->link;

printf("the deleted element is %d \n",curr->data);

free(curr);

}

}

void display()

{

if(front==NULL)

printf("queue is empty \n");

curr=front;

while(curr!=NULL)

{

printf("%3d",curr->data);

printf("->");

curr=curr->link;

}

printf("null");

}

Exercise 8:

a)write a C program that uses functions to create a single linked list.

#include<stdio.h>

#include<conio.h>

void create();

void display();

typedef struct sll

{

int data;

struct sll \*link;

}node;

node \*curr,\*root,\*newnode;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.display \n");

printf("3.exit\n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

display();

break;

default :

exit(0);

}

}

}

void create()

{

int x;

printf("enter the elements of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->link=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

printf("%3d",curr->data);

printf("->");

curr=curr->link;

}

printf("null");

}

b)write a C program that uses functions to perform insertion operation on a single linked list.

#include<stdio.h>

#include<conio.h>

void create();

void display();

void insertatbegin();

void insertatpos();

void insertatlast();

void reverse();

typedef struct sll

{

int data;

struct sll\*link;

}node;

node \*curr,\*root,\*newnode;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.display \n");

printf("3.insertatbegin \n");

printf("4.insertatpos \n");

printf("5.insertatlast \n");

printf("6.reverse\n");

printf("7.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

insertatbegin();

break;

case 4:

insertatpos();

break;

case 5:

insertatlast();

break;

case 6:

reverse();

display();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->link=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void reverse()

{

node \*p1,\*p2,\*p3;

p1=root;

p2=p1->link;

p3=p2->link;

p1->link=NULL;

p2->link=p1;

while(p3!=NULL)

{

p1=p2;

p2=p3;

p3=p3->link;

p2->link=p1;

}

root=p2;

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->link;

}

printf("NULL");

}

void insertatbegin()

{

int x;

printf("enter the element of node \n");

scanf("%d",&x);

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=newnode;

else

{

curr=root;

newnode->link=curr;

root=newnode;

}

}

void insertatpos()

{

int pos,x,c;

printf("enter the pos of node to insert \n");

scanf("%d",&pos);

printf("enter the element of node");

scanf("%d",&x);

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

newnode->data=x;

newnode->link=NULL;

curr=root;

c=1;

while(1)

{

if(c==pos-1)

break;

else

{

curr=curr->link;

c++;

}

}

newnode->link=curr->link;

curr->link=newnode;

}

void insertatlast()

{

int x;

printf("enter element of node");

scanf("%d",&x);

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

newnode->data=x;

newnode->link=NULL;

curr=root;

while(curr->link!=NULL)

{

curr=curr->link;

}

curr->link=newnode;

}

c)write a C program that uses functions to perform deletion operation on a single linked list.

#include<stdio.h>

#include<conio.h>

void create();

void display();

void deleteatbegin();

void deleteatpos();

void deleteatlast();

typedef struct sll

{

int data;

struct sll\*link;

}node;

node \*curr,\*root,\*newnode,\*prev;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.display \n");

printf("3.deleteatbegin \n");

printf("4.deleteatpos \n");

printf("5.deleteatlast \n");

printf("6.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

deleteatbegin();

break;

case 4:

deleteatpos();

break;

case 5:

deleteatlast();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->link=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->link;

}

printf("NULL");

}

void deleteatbegin()

{

curr=root;

root=root->link;

free(curr);

}

void deleteatpos()

{

int x;

curr=root;

printf("enter the element of node to delete\n");

scanf("%d",&x);

prev=NULL;

while(1)

{

if(curr->data==x)

break;

else

{

prev=curr;

curr=curr->link;

}

}

prev->link=curr->link;

free(curr);

}

void deleteatlast()

{

curr=root;

while(curr->link!=NULL)

{

prev=curr;

curr=curr->link;

}

prev->link=NULL;

free(curr);

}

Exercise 9:

a)write a C program to merge two single linked lists.

#include<stdio.h>

#include<conio.h>

void create();

void create1();

void merge();

void display();

void display1();

typedef struct sll

{

int data;

struct sll\*link;

}node;

node \*curr,\*root,\*newnode;

typedef struct sll1

{

int data1;

struct sll1\*link1;

}node1;

node \*curr1,\*root1,\*newnode1;

main()

{

int ch;

clrscr();

root=NULL;

while(1)

{

printf("\n1.create \n");

printf("2.create1 \n");

printf("3.merge \n");

printf("4.display \n");

printf("5.display1 \n");

printf("6.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

create1();

break;

case 3:

merge();

break;

case 4:

display();

break;

case 5:

display1();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->link=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void create1()

{

int x1;

printf("enter the element of node to insert \n");

scanf("%d",&x1);

while(x1!=-999)

{

newnode1=(node1\*)malloc(sizeof(node1));

newnode1->data1=x1;

newnode1->link1=NULL;

if(root1==NULL)

root1=curr1=newnode1;

else

{

curr1->link1=newnode1;

curr1=newnode1;

}

printf("another element to insert");

scanf("%d",&x1);

}

}

void merge()

{

if(root==NULL)

root=root1;

else

{

curr=root;

while(curr->link!=NULL)

curr=curr->link;

curr->link=root1;

}

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

printf("%3d",curr->data);

printf("->");

curr=curr->link;

}

printf("NULL");

}

void display()

{

curr=root;

if(root==NULL)

printf("list is underflow \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->link;

}

printf("NULL");

}

void display1()

{

if(root1==NULL)

printf(" the is underflow \n");

curr1=root1;

while(curr1!=NULL)

{

printf("%d",curr1->data);

printf("->");

curr1=curr1->link;

}

printf("null");

}

b)Write a C program to reverse elements of single linked list?

#include<stdio.h>

#include<conio.h>

void create();

void display();

void reverse();

typedef struct sll

{

int data;

struct sll\*link;

}node;

node \*curr,\*root,\*newnode;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.reverse \n");

printf("3.display\n");

printf("4.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

reverse();

display();

break;

case 3:

display();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->link=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->link=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void reverse()

{

node \*p1,\*p2,\*p3;

p1=root;

p2=p1->link;

p3=p2->link;

p1->link=NULL;

p2->link=p1;

while(p3!=NULL)

{

p1=p2;

p2=p3;

p3=p3->link;

p2->link=p1;

}

root=p2;

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->link;

}

printf("NULL");

}

c)write a C program to store a polynomial expression in memory using linked lists?

#include<stdio.h>

#include<stdlib.h>

typedef struct poly

{

int coeff;

int exp;

struct poly \*link;

}node;

node \*first,\*x;

node \*getnode()

{

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

if(x==NULL)

{

printf("Cannot allocate memory\'t");

}

return x;

}

void display()

{

node \*temp;

if(first==NULL)

{

printf("LIST EMPTY\n");

return;

}

printf("The Polynomial is\n");

temp=first;

while(temp)

{

printf("%dX%d+",temp->coeff,temp->exp);

temp=temp->link;

}

printf("\n");

}

void main()

{

node \*p;

int ch,i=1,n;

int c, e;

clrscr();

printf("Enter the no of terms in the polynomial");

scanf("%d",&n);

while(i<=n)

{

printf("coefficient and the exponent : %d :",i);

scanf("%d%d",&c,&e);

i++;

if(first)

{

p=first;

while(p->link)

p=p->link;

p->link=getnode();

p=p->link;

p->coeff=c;

p->exp=e;

p->link=NULL;

}

else

{

first=getnode();

first->coeff=c;

first->exp=e;

first->link=NULL;

}

}

display();

getch();

}

d)rite a C program that uses functions to perform insertion operation on a double linked list.

#include<stdio.h>

#include<conio.h>

void create();

void display();

void insertatbegin();

void insertatpos();

void insertatlast();

typedef struct dll

{

int data;

struct dll\*prev;

struct dll\*next;

}node;

node \*curr,\*root,\*newnode;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.display \n");

printf("3.insertatbegin \n");

printf("4.insertatpos \n");

printf("5.insertatlast \n");

printf("6.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

insertatbegin();

break;

case 4:

insertatpos();

break;

case 5:

insertatlast();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->prev=NULL;

newnode->next=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->next=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->next;

}

printf("NULL");

}

void insertatbegin()

{

int x;

printf("enter the element of node \n");

scanf("%d",&x);

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

newnode->data=x;

newnode->prev=NULL;

newnode->next=NULL;

if(root==NULL)

root=newnode;

else

{

curr=root;

newnode->next=curr;

curr->prev=newnode;

root=newnode;

}

}

void insertatpos()

{

int pos,x,c;

printf("enter the pos of node to insert \n");

scanf("%d",&pos);

printf("enter the element of node");

scanf("%d",&x);

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

newnode->data=x;

newnode->prev=NULL;

newnode->next=NULL;

curr=root;

c=1;

while(1)

{

if(c==pos-1)

break;

else

{

curr=curr->next;

c++;

}

}

curr->next->prev=newnode;

newnode->next=curr->next;

newnode->prev=curr;

curr->next=newnode;

}

void insertatlast()

{

int x;

printf("enter element of node");

scanf("%d",&x);

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

newnode->data=x;

newnode->prev=NULL;

newnode->next=NULL;

curr=root;

while(curr->next!=NULL)

{

curr=curr->next;

}

curr->next=newnode;

newnode->prev=curr;

}

.pa

e)write a C program that uses functions to perform deletion operation on a double linked list.

#include<stdio.h>

#include<conio.h>

void create();

void display();

void deleteatbegin();

void deleteatpos();

void deleteatlast();

typedef struct dll

{

int data;

struct dll\*prev;

struct dll\*next;

}node;

node \*curr,\*root,\*newnode,\*prev;

int x;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create \n");

printf("2.display \n");

printf("3.deleteatbegin \n");

printf("4.deleteatpos \n");

printf("5.deleteatlast \n");

printf("6.exit \n");

printf("enter your choice \n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

deleteatbegin();

break;

case 4:

deleteatpos();

break;

case 5:

deleteatlast();

break;

default :

exit(0);

}

}

getch();

}

void create()

{

int x;

printf("enter element of node to insert \n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->prev=NULL;

newnode->next=NULL;

if(root==NULL)

root=curr=newnode;

else

{

curr->next=newnode;

curr=newnode;

}

printf("another element to insert \n");

scanf("%d",&x);

}

}

void display()

{

curr=root;

if(root==NULL)

printf("list is empty \n");

while(curr!=NULL)

{

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

printf("->");

curr=curr->next;

}

printf("NULL");

}

void deleteatbegin()

{

curr=root;

root=root->next;

curr->next->prev=NULL;

printf("the deleted node is %d \n",curr->data);

free(curr);

}

void deleteatpos()

{

node \*t;

t=NULL;

curr=root;

printf("enter the element of node to delete\n");

scanf("%d",&x);

while(1)

{

if(curr->data==x)

break;

else

{

t=curr;

curr=curr->next;

}

}

curr->next->prev=curr;

t->next=curr->next;

printf("the deleted element is %d \n",curr->data);

free(curr);

}

void deleteatlast()

{

node \*t;

curr=root;

while(curr->next!=NULL)

{

t=curr;

curr=curr->next;

}

t->next=NULL;

printf("the deleted element is %d \n",curr->data);

free(curr);

}

Exercise 10:

a)write a C program to create a binary tree of integers?

b)write a recursive c program ,for traversing a binary tree in preorder,inorder and postorder.

#include<stdio.h>

#include<conio.h>

typedef struct BT

{

int data;

struct BT \*lchild,\*rchild;

}node;

node \*root,\*newnode,\*curr,\*prev;

int x;

void inorder(node \*);

void preorder(node \*);

void postorder(node \*);

void create(node \*);

void main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create\n");

printf("2.inorder\n");

printf("3.preorder\n");

printf("4.postorder\n");

printf("5.exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\nenter root(type -999 to stop):");

scanf("%d",&x);

if(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

if(root==NULL)

{

root=newnode;

create(root);

}

}

break;

case 2:inorder(root);

break;

case 3:preorder(root);

break;

case 4:postorder(root);

break;

default:exit(0);

}

}

getch();

}

void create(node \*curr)

{

prev=curr;

printf("\n enter left child of%d",prev->data);

scanf("%d",&x);

if(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

prev->lchild=newnode;

prev=newnode;

create(prev);

}

prev=curr;

printf("\ enter right child of%d",prev->data);

scanf("%d",&x);

if(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

prev->rchild=newnode;

prev=newnode;

create(prev);

}

}

void inorder(node \*curr)

{

if(curr!=NULL)

{

inorder(curr->lchild);

printf("%3d",curr->data);

inorder(curr->rchild);

}

}

void preorder(node \*curr)

{

if(curr!=NULL)

{

printf("%3d",curr->data);

preorder(curr->lchild);

preorder(curr->rchild);

}

}

void postorder(node \*curr)

{

if(curr!=NULL)

{

postorder(curr->lchild);

postorder(curr->rchild);

printf("%3d",curr->data);

}

}

c)write a nonr-recursive c program ,for traversing a binary tree in preorder,inorder and postorder.

#include<stdio.h>

typedef struct btree

{

int data;

struct btree \*lchild, \*rchild;

} node;

node \*root, \*newnode,\*prev,\*curr,\*stack[30];

int x,top=-1;

void nonrecinorder();

void nonrecpreorder();

void nonrecpostorder();

void create(node \*);

void push(node \*);

node \*pop();

void main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create\n");

printf("2.nonrecinorder\n");

printf("3.nonrecpreorder\n");

printf("4.nonrecpostorder\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n Enter Root (type -999 to stop): ");

scanf("%d", &x);

if (x != -999)

{

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

newnode->data = x;

newnode->lchild = newnode->rchild = NULL;

if (root == NULL)

{

root = newnode;

create (root);

}

}

break;

case 2:

nonrecinorder();

break;

case 3:

nonrecpreorder();

break;

case 4:

nonrecpostorder();

break;

default:

exit(0);

}

}

getch();

}

void create(node \*curr)

{

prev = curr;

printf( "\n Enter Left Child of %d (type -999 to stop): ",prev->data);

scanf("%d",&x);

if (x != -999)

{

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

newnode->data = x;

newnode->lchild = newnode->rchild = NULL;

prev->lchild = newnode;

prev = newnode;

create(prev);

}

prev = curr;

printf("\n Enter Right Child of %d (type -999 to stop):",prev->data);

scanf("%d",&x);

if (x != -999)

{

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

newnode->data = x;

newnode->lchild = newnode->rchild = NULL;

prev->rchild = newnode;

prev = newnode;

create(prev);

}

}

void push (node \*t)

{

top++;

stack[top] = t;

}

node \*pop ()

{

node \*t;

t = stack[top];

top--;

return t;

}

void nonrecinorder(void)

{

curr = root;

printf( "\n Inorder is:");

while (curr != NULL)

{

while (curr != NULL)

{

if (curr->rchild != NULL)

push (curr->rchild);

push (curr);

curr = curr->lchild;

}

curr = pop ();

while ((!(top == -1)) && (curr->rchild == NULL))

{

printf("%3d",curr->data);

curr = pop ();

}

printf("%3d", curr->data);

if (!(top == -1))

curr = pop ();

else

curr = NULL;

}

}

void nonrecpreorder()

{

curr = root;

printf("\n Preorder is:");

if (curr != NULL)

{

push (curr);

while (!(top == -1))

{

curr = pop ();

printf("%3d",curr->data);

if (curr->rchild != NULL)

push (curr->rchild);

if (curr->lchild != NULL)

push (curr->lchild);

}

}

}

void nonrecpostorder (void)

{

curr = prev = root;

printf("\n Postorder is:");

while (curr != NULL)

{

for (; curr->lchild != NULL; curr=curr->lchild)

push (curr);

while (curr!=NULL && (curr->rchild==NULL) || (curr->rchild==prev))

{

printf("%3d",curr->data);

prev = curr;

if (top == -1)

return;

curr = pop ();

}

push (curr);

curr = curr->rchild;

}

}

Exercise 11:

a)write a c program to create a BST.

#include<stdio.h>

#include<conio.h>

void create();

void insert();

void inorder();

typedef struct BST

{

int data;

struct BST \*lchild;

struct BST \*rchild;

}node;

node \*root,\*curr,\*newnode,\*pre;

main()

{

int ch;

clrscr();

while(1)

{

printf("\n1.create\n");

printf("2.insert\n");

printf("3.inorder\n");

printf("4.exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:create();

break;

case 2:insert();

break;

case 3:inorder();

break;

default:exit(0);

}

}

}

void create()

{

int x;

printf("enter element of node to insert\n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

if(root==NULL)

curr=root=newnode;

else

{

curr=root;

while(curr!=NULL)

if(curr->data<newnode->data)

{

pre=curr;

curr=curr->rchild;

}

else

{

pre=curr;

curr=curr->lchild;

}

if(pre->data<newnode->data)

pre->rchild=newnode;

else

pre->lchild=newnode;

}

scanf("%d",&x);

}

}

void insert(int x,node \*curr)

{

printf("enter element of node to insert\n");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

if(root==NULL)

curr=root=newnode;

else

{

if(curr==NULL)

{

curr=newnode;

{

if(x<curr->data)

insert(x,curr->lchild);

else if(x>curr->data)

insert(x,curr->rchild);

else

printf("the element is duplicate:do nothing");

}

}

}

}

}

void inorder()

{

if(curr!=NULL)

{

inorder(curr->lchild);

printf("%3d",curr->data);

inorder(curr->rchild);

}

}

b)write a C program to insert a node into a BST.

#include<stdio.h>

#include<conio.h>

typedef struct BST

{

int data;

struct BST \*lchild;

struct BST \*rchild;

}node;

node \*root,\*curr,\*newnode,\*prev;

int x;

void create();

void insert();

void inorder(node \*);

void main()

{

int ch;

clrscr();

while(1)

{

printf("\n 1.create\n");

printf("\n 2.insert\n");

printf("\n 3.exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:create();

break;

case 2:insert();

break;

default :exit(0);

}

}

getch();

}

void create()

{

printf("enter the values into bst");

scanf("%d",&x);

while(x!=-999)

{

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

if(root==NULL)

curr=root=newnode;

else

{

curr=root;

while(curr!=NULL)

if(curr->data<newnode->data)

{

prev=curr;

curr=curr->rchild;

}

else

{

prev=curr;

curr=curr->lchild;

}

if(prev->data<newnode->data)

prev->rchild=newnode;

else

prev->lchild=newnode;

}

scanf("%d",&x);

}

printf(" inorder is \n");

inorder(root);

}

void insert()

{

printf("enter an element to insert");

scanf("%d",&x);

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

newnode->data=x;

newnode->lchild=newnode->rchild=NULL;

if(root==NULL)

curr=root=newnode;

else

{

curr=root;

while(curr!=NULL)

if(curr->data<newnode->data)

{

prev=curr;

curr=curr->rchild;

}

else

{

prev=curr;

curr=curr->lchild;

}

if(prev->data<newnode->data)

prev->rchild=newnode;

else

prev->lchild=newnode;

}

printf("\n inorder after entering elements is ");

inorder(root);

}

void inorder(node \*curr)

{

if(curr!=NULL)

{

inorder(curr->lchild);

printf("%3d",curr->data);

inorder(curr->rchild);

}

}

Exercise 12:

a)Write a C program to compute the shortest path of a graph using Dijkstra`s Algorithm?

#include <stdio.h>

#include<conio.h>

#define GRAPHSIZE 10

#define INFINITY 1000

#define MAX(a, b) ((a > b) ? (a) : (b))

int e; /\* The number of nonzero edges in the graph \*/

int n; /\* The number of nodes in the graph \*/

long dist[GRAPHSIZE][GRAPHSIZE]; /\* dist[i][j] is the distance between node i and j; or 0 if there is no direct connection \*/

long d[GRAPHSIZE]; /\* d[i] is the length of the shortest path between the source (s) and node i \*/

void printD() {

int i;

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

printf("%10d", i);

printf("\n");

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

printf("%10ld", d[i]);

}

printf("\n");

}

void dijkstra(int s) {

int i, k, mini;

int visited[GRAPHSIZE];

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

d[i] = dist[i][s];

visited[i] = 0; /\* the i-th element has not yet been visited \*/

}

d[s] = 0;

visited[s]=1;

for (k = 1; k <= n; ++k) {

mini = -1;

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

if (visited[i]==0 && ((mini == -1) || (d[i] < d[mini])))

mini = i;

visited[mini] = 1;

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

if (visited[i]==0)

if (d[mini] + dist[mini][i] < d[i])

d[i] = d[mini] + dist[mini][i];

}

}

int main(void) {

int i, j;

int u, v, w;

printf("Ente the No of Edges");

scanf("%d", &e);

printf("Ente the No of Nodes");

scanf("%d", &n);

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

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

dist[i][j] = INFINITY;

for (i = 0; i < e; ++i) {

scanf("%d%d%d",&u,&v,&w);

dist[u][v] = dist[v][u]=w;

}

dijkstra(1);

printD();

return 0;

}

b)Write a C program to find the minimum spanning tree using Warshall`s algorithm?

#include<stdio.h>

#include<conio.h>

void display( int matrix[ 20 ][ 20], int n )

{

int i, j;

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

{

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

printf( "%5d", matrix[ i ][ j ] );

printf( "\n" );

}

}

int minimum( int a, int b )

{

if ( a <= b )

return a;

else

return b;

}

void main()

{

int i, j, k, n;

int w[ 20 ][ 20 ], a[ 20 ][ 20];

clrscr();

printf( "Enter number of vertices : " );

scanf( "%d", &n );

printf( "Enter weighted matrix :\n" );

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

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

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

printf( "Weighted matrix is :\n" );

display( w, n );

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

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

a[ i ][ j ] = w[ i ][ j ];

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

{

printf( "\nA%d is :\n", k );

display( a, n );

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

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

a[ i ][ j ] = minimum( a[ i ][ j ] , a[ i ][ k ] + a[ k ][ j ] );

}

printf( "Shortest path matrix A%d is :\n", k );

display( a, n );

}