List->[3|]->[10|]->[7|/]

a.) insert 20 in front of the linked list

List->[3|]->[10|]->[7|/]

P->[20|/]//connect to list before 3

node \*p

p=new(node);

p->info=20;

i.)p->next=list

ii.)list=p;

b.) insert 30 at the rear of the list

List->[20|]->[3|]->[10|]->[7|/]

P->[30|/]

p=new(node);

p->info=30; p->next=NULL;

//we need another pointer to point to the last node we declare q but first we //make it first point to first node to visit all nodes

node \*q=list;

//when last pointer points to NULL

while(q->next!=NULL)

{

q = q->next;

}//end while

q->next=p; //this attaches 30 to the rear or after the 7 which we wanted

c.) delete the first node

List->[10|]->[5|]->[11|/]

p=list;

i.)list=p->next

ii.) delete(p); //we get rid of the node but pointer p still exists

note the following are the same

p=new(node); -> p = new node;

delete(p); -> delete p;

d.) delete the last node

List->[5|]->[11|]->[6|/]

//if we want to delete something we need a pointer to point to that

//for this one we’ll need 2 pointers so example we have p and q

//we want q to follow p bec we want q to point to where p is at

while(p->next!=NULL;)

{

q=p;

p=p->next;

}//end while

q->next= NULL

delete p;

List->[5|]->[11|/]

e.) insert 10 before node whose info = 20;

List->[30|]->[5|]->[70|]----->[20|]->[100|]

p->[10|] //for this we’ll need two connectors

//we’ll need a pointer to point to 20s address let’s use r and a pointer to //point to 70 we’ll use q

//HOW TO FIND q AND r?

//same situation q has to follow r

q=r=list;

while(r->info!=20)

{

q=r;

r=r->next;

}//endwhile

//let’s see trace

r and q are both pointing to first node

is r info==20? no so q is equal to r and now r is r->next

back to loop, is r info ==20? no it’s 5 so now q=r and r is r->next

is r info==20? no it’s 70 so now q=r and r is r->next

is r info==20? yes so break out of loop while q is still in info 70

1. q->next=p; //bec we want to insert p with info 10 after q which is before r
2. p->next=r;

ordered linked list to sort out anything you put in the list

List->[5|]->[10|]------>[30|]->[50|]

INSERT 20!

q=r=list; // List->[5|]->[10|]------>[30|]->[50|]

q r

//same as above while except while condition changes

while(r->info<20)

{

q=r;

r=r->next;

}//endwhile

//let’s see trace

r and q are both pointing to first node

is r info < 20? yes enter loop so q is equal to r and now r is r->next

back to loop, is r info < 20? yes it’s 10 so now q=r and r is r->next

is r info < 20? no it’s 70 so now q=r and r is r->next

q->next=p;

p->next=r;

f.) Given the following ordered linked list

delete 30

List->[5|]->[10|]->[20|]->[30|]->[50|]->[60|]

//let’s have pointer r and point it to 30 and we want to connect address in 20 and 50 connect so we’ll need another pointer to point to 20 let’s use p

so we’ll have p->next = r->next

p=r=list;

//as long as r->info is !=30 we want p to follow r

while(r->info!=30)

{

p=r; r=r->next;

}//endwhile

//this one order matters we want to connect it first before we del r

1. p->next = r->next;
2. delete r;

------------APPLICATION-------------

if we have example a string

Mom //we push it little by little into linked list implementation of stack and implementation of queue PALINDROME STUFF

srack->[/] queue->[/] //pointing to NULL

pushs(‘M’); pushq(‘M’);

stack->[M|/] queue->[M|/]

pushs(‘o’); pushq(‘o’);

//first item u pop last item u push for stack?

stack->[o|]->[M|/] queue->[M|]->[o|/] //first in first out

pushs(‘m’); pushq(‘m’);

stack->[m|]->[o|]->[M|/] queue->[M|]->[o|]->[m|/]

class STACK

{. . .}

class QUEUE

{. . .}

int main()

{

QUEUE<char> q; //q->[/] or NULL

STACK<char> s; //s->[/] or NULL

char c;

//read the input char by char and push (we don’t care bout char or

//spaces only letters) letter in stack and queue

while(cin.get©, c!=’\n’)

{

if(isalpha())//to eliminate spaces and special chars

{

c=toupper(c);

q.push\_q(c); s.push\_s(c);

}//endif

}//endwhile

//check for palindrome

char c1, c2;

while(!q.empty\_q())

{

c1=q.pop\_q();

c2=s.pop\_s();

if(c1 != c2)

{break; //if not break out of this loop}

}//endwhile

if(q.empty\_q())

{cout<<”u have palindrome\n”;}

else{cout<<”not palindrome\n”;}

}//endmain

//----------------example 2------------------

write a program to find the value of a function whose coefficient and the power of each term is given

Example:

f(x)=(4x^3)-(3x^2)+(2x)-(10x^0)

f->[4|3| ]->[-3|2| ]->[2|1| ]->[-10|0|/] //every new term is at the rear

//for this we’ll need a room for the coefficient, the power, and the next pointer.

FIND VALUE OF f at x=10

if p is pointing to first node

value=p->coef\*pow(10,p->pwr);

struct node

{

int coef;

int pwr;

node \*next;

}//end structnode

#include <iostream>

using namespace std

template <class T>

class OrderedLinkedList

{

private:

struct NODE

{

T info;

NODE \*next;

};//end struct node

NODE \*list;

public:

OrderedLinkedList(){list=NULL;}

void insert(T x)

{

NODE \*p=list, \*q=list, \*r;

//create a new node

r=new(NODE); r->info=x;

r->next=NULL;

//find insertion place

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

{

q=p; p=p->next;

}//endwhile

if(p==list) //x is the first info

{

list r; r->next=p;

}//endif

else if(p==NULL)//x is the last info

{

q->next=r;

}//endelseif

else //x is neither first nor last info

{

r->next=p; q->next=r;

}//endelse

}//end insert

void display()

{

NODE \*p=list;

while(p!=NULL)

{

cout<<p->info<<”->”; p=p->next;

}//endwhile

cout<<”NULL”<<endl;

}//end display

}//end orderedlinkedlist

int main()

{

//create a set of integers

OrderedLinkedList<int> setA;

//insert 20,5,11,9 in setA

setA.insert(20). etc. . .

//display setA

setA.display();

}//end main