Exam no. 1

This wed

Projects 1-4 :O

cout<<atoi(”125”)+5; //changes string to an integer = 130

string st=”125”;

cout<<stoi(st)+10; // ~~stoi(st)~~ 🡪 125+10 = 135

//switch statements are only good for char types or int types not strings

//-----------for proj 5[prob 1---------------------

Creat the following linked list using for loopp

p->[1 |]->[2|]->[3|]->[4|]

struct node

{

int info;

node \*next;

};

main

node \*p;

//new lets them find me a new room in memory

//(node) //a location big enough to store this info

p=new(node);

//so far we’ve created p->[ ]

p->info=1; //p->[1//info | //next]

//remember p is the master key so we made a copy

node \*r;

r=p; //r is now pointing to the same thing as p

for(int i=2; i<=4; ++i) //bec those are numbs we putting in nodes 2-4

{

//find a new location to store this info

r->next=new(node); //r->[info|next] 🡪 r=r->next[info\next]

r=r->next;

r->info=I //if u have something to read all time do cin>>r->info

//like example r->age, r->name. . . etc.

}

what happens in that operation above

p->[1 | r=r->next]r->//r becomes this now[2 |r=r->next]r->[3 |r=r->next]r-> [4 |] //the last next is missing, so we write after exiting the loop

r->//same as p above now

r->next=NULL; //you mark it as 0

//if we want to start over we can go back to p to go through it again

//-------display all nodes------------

r=p; //p I sthe master key, always pointing to first

while(r!=NULL)//always while to do this //while not NULL bec finished

{

cout<<r->info<<”->”;

//we want our output to be 1->2->3->4->NULL

r=r->next;

}

cout<<”NULL\n”;

//output = 1->2->3->4->NULL;

EX: Data: 3 9 7

//linked list is gonna look like this

insert each new number in Front of the list

|  |
| --- |
| 7 |
| 9 |
| 3 |

//insert 3 p->[3 | ]

//insert 9 p->[9 | ]->[3 | ]

//insert 7 p->[7 | ]->[9 | ]->[3 |/]

//if we visit we will see 7 9 3

//the relationship is they are reversed

//if u want a linked list to behave like stack, every new data must be inserted in front of the list

//the other name for stack is last-in-first out (LIFO)

//now we want to insert data: 3 9 7 into a linked list to behave like queue

//a queue is like first person in line is first person to get service

//first-in-first-out (FIFO)

//insert 3 p->[3 | ]

//insert 9 p->[3 | ]->[9 | ]

//insert 7 p->[3 | ]->[9 | ]->[7 |/]

//if we visit we will see 3 9 7

QUEUE and STACK have a lot of implications in programming

Ex. Given the following list

pp->[10|]->[15|]->[7|/]

front rear

insert 30 in front of the list

-create a new node

node \*q=new(node);

q->[30|]

q->info=30; //do not lose master key

//p->[10|]->[15|]->[7|]

//q->[30 ]->?

//q->[30|q->next]->

// we want to connect the q->next 30 to the p. if p was a10 we want [30|a10]

//q->[30|p] q->next=p;

//now we want p to point to q

//i.) q->next=[;

//ii.) p=q; //so now q connected to p but we want now p to connect to q which //connects all of them back together

Ex 2. t->[Tom|]->[Bob|]->[Mary|/]

write statement to insert “John” at the front of the list

t->[John|]->[Tom|]->[Bob|]->[Mary|/]

a100 a150 a130

t=a100 t->[Tom|150]->[Bob|130]->[Mary|0]

i.) create a new node

s=new(node)

//now we have this s->[ ]

a200

s->name=”John”

//now we have s->[John| ] but now we want to connect s to a100 which was t

//the order of connections matter bec u can lose their connections

ii.) s->next=t;

iii.) t=s; //t is now == to a200

Ex. 3 t->[Tom|]->[Bob|]->[Mary|/]

write statement to insert “Adam” at the rear of the list

t= t->[Tom|]->[Bob|]->[Mary|]->[Adam|/]

t->[Tom|]->[Bob|]->[Mary|/]

i.) r=new(node) //now we have r->[ |]

r->name=”Adam”; //r->[Adam|]we want this to be the last node

r->next=NULL; //r->[Adam|/]

//we now want this connection t->[Tom|]->[Bob|]->[Mary|]->[Adam|/]

//let’s declare p a copy of t to look for the last

//we go through a loop to check a condition to see if the last node is null

ii.) to find p to point to the last node

p=t;

while(p->next!=NULL)

{

p=p->next;

} // p->[Tom|]->[Bob|]->[Mary|/] //once p points to mary then it is null

//we then connect r to the address of p. suppose r was a500 and we want to //put it in p’s last pointing p->next

iii.) connect points

p->next=r;

#include <iostream>

#include <string>

using namespace std;

int main()

{

string days[5]={“Mon”, “Tue”,. . .”Fri”}

struct node

{

string name;

node \*next;

};

node \*t;

t=new(node);

//t->[Mon|]

t->name=days[0];

t->next=NULL;

//since we used Monday we start with i=1

for(int i=1;i<5;++i)

{

//create a new node

r=new(node);

r->name=days[i];

//t->[Mon/] and we want r->[Tue| ] to connect to t

//connect pointers

//i.)

r->next=t;

//ii.)

t=r;

}//end for

//result will be t->[Fri|]->[Thu|]->[Wed|]. . .etc

}//end main