CS 16 5/21

**Recursion**

* Fractals are examples of recursion.
* The last recursive call is called the base case.
* Otherwise, we are in the recursive case which is solving a smaller subproblem of the larger problem.

Sample File:

#include <iostream>

#include <string>

using namespace **std**;

struct LLNode{

 int data;

 LLNode\* nextNode;

}; *// a mode of a linked list*

struct LL{

 LLNode\* head;

 LLNode \*tail;

}; *// a linked list*

int **sumOfLinkedList**(LLNode\* *l*){

 if(l->nextNode == NULL)

 {

 return l->data;

 }

 else{

 *// RECURSIVE CASE*

 return l->data + **sumOfLinkedList**(l->nextNode);

 }

}

int **main**()

{

 *// Create a linked list: (123,456,567,789) then change it to (115,123,234,456,567,737,789,999)*

 LLNode \*head = new LLNode; *// This keeps track of the start of the list*

 LLNode \*zero = new LLNode;

 LLNode\* one = new LLNode;

 LLNode \* two = new LLNode;

 LLNode \*three = new LLNode;

 LLNode\* tail = new LLNode;

 *// First set the data for all the elements*

 zero->data = 123;

 one->data = 456;

 (\*two).data = 567; *// Notice we can use the -> or (\*). to access*

 three->data = 789;

 *// Now lets connect the linked list*

 head->nextNode = zero;

 (\*zero).nextNode = one;

 one->nextNode = two;

 two->nextNode = three;

 three->nextNode = NULL; *// This one is the end of the list so we set it to NULL*

 tail-> nextNode = three;

 *//Now lets modify the linked list to be (115,123,234,456,567,737,789,999)*

 LLNode\* aNode0 = new LLNode; *// This will later store 115*

 LLNode\* aNode1 = new LLNode; *// This will later store 234*

 LLNode\* aNode2 = new LLNode; *// This will later store 737*

 LLNode\* aNode3 = new LLNode; *// This will later store 999*

 *// We could set the nextNode first, but we're doing it this way to better follow the logic*

 aNode0->data = 115; *// Now this stores 115*

 aNode1->data = 234; *// Now this stores 234*

 aNode2->data = 737; *// Now this stores 737*

 aNode3->data = 999; *// Now this stores 999*

 *// Now set the head to 115 instead*

 head->nextNode = aNode0;

 aNode0->nextNode = zero; *// Set the node we are inserting's next node to the node after head*

 *// Now put 234 in between 123 (zero) and 456 (one)*

 aNode1->nextNode = zero->nextNode; *// Node after 234 is 456 now*

 zero->nextNode = aNode1; *// Node after 123 is 234 now*

 *// Now put 737 in between 567 (two) and 789 (three)*

 aNode2->nextNode = two->nextNode; *// Node after 737 is 789 now*

 two->nextNode = aNode2; *// Node after 567 is 737 now*

 *// Now put 999 after 789 (three)*

 three->nextNode = aNode3; *// Node after 789 is 999 now*

 aNode3->nextNode = NULL; *// 999's next node indicates the end of the linked list*

 LL\* list = new LL;

 list->head = head;

 list->tail = tail;

 cout << "Printing the linked list: \n\n";

 for(LLNode\* i = head->nextNode; i != NULL; i = i->nextNode){

 if(i == head->nextNode) cout << "Linked list: ";

 cout << i->data << "->";

 if(i->nextNode == NULL) cout << 'X';

 }

 cout << "\n\n";

 *// Sum should be 115+123+234+456+567+737+789+999 = 4020*

 cout << "Sum of the linked list is " << **sumOfLinkedList**(list->head) << "\n\n";

 return 0;

}

Output:

Printing the linked list:

Linked list: 115->123->234->456->567->737->789->999->X

Sum of the linked list is 4020

* Notice that the only recursive part of this is the function sumOfLinkedList(). This takes the first node in the linked list as the parameter and propagates through the linked list from there.
* Alternatively you could use this function to sum a contiguous subset which concludes with the tail of the linked list.

**Helper Functions**

* Sometimes recursion can be tricky
* This makes us want more abstraction
* We don’t have to think the whole problem through at once, so we do it in parts with helper functions

Sample file of helper function:

#include <iostream>

#include <string>

using namespace **std**;

*// STRUCT AND FUNCTION DECLARATIONS (definitions are below main)*

struct LLNode{

 int data;

 LLNode\* nextNode;

}; *// a mode of a linked list*

struct LL{

 LLNode\* head;

 LLNode \*tail;

}; *// a linked list*

int **sumOfLinkedList**(LLNode\* *p*);

int **sumMyList**(LL\* *l*);

void **printLL**(LL\**l*);

int **main**()

{

 *// Create a linked list: (123,456,567,789) then change it to (115,123,234,456,567,737,789,999)*

 *// YOU SHOULD IGNORE THE FOLLOWING UNTIL THE FUNCTION CALL: printLL(list);*

 LL\* list = new LL;

 LLNode \*zero = new LLNode;

 LLNode\* one = new LLNode;

 LLNode \* two = new LLNode;

 LLNode \*three = new LLNode;

 LLNode\* tail = new LLNode;

 LLNode\* aNode0 = new LLNode;

 LLNode\* aNode1 = new LLNode;

 LLNode\* aNode2 = new LLNode;

 LLNode\* aNode3 = new LLNode;

 aNode0->data = 115;

 aNode1->data = 234;

 aNode2->data = 737;

 aNode3->data = 999;

 zero->data = 123;

 one->data = 456;

 (\*two).data = 567;

 three->data = 789;

 list->head = zero;

 (\*zero).nextNode = one;

 one->nextNode = two;

 two->nextNode = three;

 three->nextNode = NULL;

 list->tail = three;

 list->head = aNode0;

 list->head->nextNode = zero;

 aNode1->nextNode = zero->nextNode;

 zero->nextNode = aNode1;

 aNode2->nextNode = two->nextNode;

 two->nextNode = aNode2;

 list->tail->nextNode = aNode3;

 list->tail = aNode3;

 aNode3->nextNode = NULL;

 *// RESUME READING HERE*

 **printLL**(list);

 *// Sum should be 115+123+234+456+567+737+789+999 = 4020*

 *// cout << "Sum of the linked list is " << sumOfLinkedList(list->head) << "\n\n";*

 *// USE THE HELPER FUNCTION*

 cout << "Sum of the linked list is " << **sumMyList**(list) << "\n\n";

 return 0;

}

int **sumOfLinkedList**(LLNode\* *p*){

 if(p->nextNode == NULL)

 {

 return p->data;

 }

 else{

 *// RECURSIVE CASE*

 return p->data + **sumOfLinkedList**(p->nextNode);

 }

}

int **sumMyList**(LL\* *l*){

 return **sumOfLinkedList**(l->head);

}

void **printLL**(LL\* *l*){

 cout << "\n\nPrinting the linked list: \n\n";

 for(LLNode\* i = l->head; i != NULL; i = i->nextNode){

 if(i == l->head) cout << "Linked list: ";

 cout << i->data << "->";

 if(i->nextNode == NULL) cout << 'X';

 }

 cout << "\n\n";

}

Output:

Printing the linked list:

Linked list: 115->123->234->456->567->737->789->999->X

Sum of the linked list is 4020

* Notice that this code uses helper function sumMyList(LL\* l);
* We also use encapsulation by defining a function printLL