**2Linked Lists**

* Given the head of a singly linked list, reverse the list, and return the reversed list. ***The function must reverse the nodes, not just the data.***

void reverse(Node\* head){

    if(head == nullptr){

        return;

    }

    Node\* prev = nullptr;

    Node\* cur = head;

    Node\* next = nullptr;

    while(cur != nullptr){

        next = cur->next;

        cur->next = prev;

        prev = cur;

        cur = next;

    }

    head = prev;

}

* Given the head of a linked list, write a recursive function to count the number of nodes in a linked list. Return the total number of nodes in the list.

int countNodes(Node\* head){

    if(head == nullptr){

        return 0;

    }

    return 1 + countNodes(head->next);

}

* You are given the heads of two sorted linked lists list1 and list2. Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list.

Node\* mergeTwoLists(Node\* list1, Node\* list2) {

    if(list1 == nullptr) return list2;

    if(list2 == nullptr) return list1;

    if(list1->val <= list2->val){

        list1->next = mergeTwoLists(list1->next, list2);

        return list1;

    }else{

        list2->next = mergeTwoLists(list1, list2->next);

        return list2;

    }

}

* Given a doubly linked list, write a function that prints the data of the nodes in the linked list backwards.

void printBackwards(){

    if(head == nullptr){

        return;

    }

    Node\* cur = head;

    while(cur->next != nullptr){

        cur = cur->next;

    }

    while(cur != nullptr){

        cout << cur->val << " ";

        cur = cur->prev;

    }

}

* Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null.

Node\* getIntersectionNode(Node \*headA, Node \*headB) {

    Node \*temp;

    while(headA != nullptr){

        temp = headB;

        while(temp != nullptr){

            if(headA == temp){

                return headA;

            }

            temp = temp -> next;

        }

        headA = headA -> next;

    }

    return nullptr;

}

* Given a circular Linked list, write a function that creates another linked list where the value of that node is the sum of original linked lists nodes excluding that index
* Given 2 linked list (passed by its head), create and return a 3rd linkedlist where the value of each node in the 3rd linked list belongs to the first 2

Node\* inBoth(Node\* head1, Node\* head2){

    Node\* solution = nullptr;

    Node\* curr1 = head1;

    Node\* curr2 = head2;

    while(curr1 != nullptr){

        while(curr2 != nullptr){

            if(curr1->data == curr2->data){

                Node\* temp = new Node();

                temp->next - solution;

                temp->data = curr1->data;

                solution = temp;

            }

            curr2 = curr2->next;

        }

        curr2 = head2;

        curr1 = curr1->next;

    }

    return solution;

}

* Other practice:  
  insert at beginning, middle, and end of a single, double, and circular linked list  
  delete at beginning, middle, and end of a single, double, and circular linked list

**Recursion**

* Write a recursive function to reverse a string. The function reverseStr() will take in a string and return the reversed string.

string reverseStr(string str){

    if (str.length() < 2){

        return str;

    }

    return reverseStr(str.substr(1)) + str.substr(0, 1);

}

* Write a recursive function that takes in a number n and returns the factorial of the n.

int factorial(int n){

    if(n < 1){

        return 1;

    }

    return n \* factorial(n-1);

}

* Write a recursive function to check if a string is a palindrome. Return a Boolean value.

bool palindrone(string str, int low, int high){

    if(low >= high){

        return true;

    }else{

        if(str[low] == str[high]){

            return palindrone(str, low+1, high-1);

        }else{

            return false;

        }

    }

}

bool palindrone(string str){

    return palindrone(str, 0, str.length()-1);

}

* Write a recursive function *for the Fibonacci Sequence.* The function fib() takes in a number n and returns the value of the n-th Fibonacci Number.  
  Ex. The sequence is as follows: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …  
  Where F(n) = F(n-1) + F(n-2)

int fib(int n){

    if(n <= 1){

        return n;

    }

    return fib(n-1) + fib(n-2);

}

* Write a recursive function that takes in a positive integer and returns the sum of its digits.

int sumOfDigits(int n){

    if(n < 10){

        return n;

    }

    return (n % 10 + sumOfDigits(n / 10));

}

* What is the output of fun1(5)?

void fun1(int x){

    if(x > 1){

        fun(--x);

        cout << --x << " ";

        fun(x-1);

    }

}

* + Answer: 0 1 2 3 0
* What is the output of fun2(5)?

void fun2(int x){

    if(x >= 1){

        fun(x-2);

        x--;

        fun(x-2);

        cout << x << " ";

    }

}

* + Answer: 0 2 1 4

**Dynamic Arrays**

* Write a program that that creates a dynamically allocated array. Assign any values to the array. And de-allocate the array.

    int maxRows, maxColumns;

    maxRows = 5; // for some int

    maxColumns = 5; // for some int

    // initiallizing 2D array

    int\*\* matrix = new int\*[maxRows];

    for (int i = 0; i < maxRows; i++) {

        matrix[i] = new int[maxColumns];

    }

    // assigning values to 2D array

    for(int i = 0; i < maxRows; i++){

        for(int j = 0; j < maxColumns; j++){

            matrix[i][j] = i; // assigning i into matrix

        }

    }

    // display 2D array

    for(int i = 0; i < maxRows; i++){

        for(int j = 0; j < maxColumns; j++){

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    // de-allocate 2D array

    for(int i = 0; i < maxRows; i++){

        delete[] matrix[i];

    }

    delete[] matrix;

* Other practice:
  + Using a class of dynamic arrays,  
    Insert at beginning, middle, and end of a dynamic array

Delete at beginning, middle, and end of a dynamic array

**Big O**

* What is the Big O of the 3 sorts?
* Bubble Sort:
  + Best: O(n^2) [for version 1]
    - Note: assume on the exam, you are given v2, unless stated otherwise
  + **Best: O(n)** [for version 2]
  + **Worst O(n^2)**
* Selection Sort:
  + **Best: O(n^2)**
  + **Worst: O(n^2)**
* Insertion Sort:
  + **Best: O(n)**
  + **Worst O(n^2)**

What is the time complexity of the following programs?

for(int i = 0; i < 100; i++){

    for(int j = 0; j < n; j++){

        cout << i << " " << j << endl;

    }

}

* + Answer: O(n) //hint: drop the constant

for(int i = 0; i >= n; i/=2){

    for(int j = m; j >= 0; j/=2){

        cout << i << " " << j << endl;

    }

}

* + Answer: O(log(n) \* log(m))

int i = 0;

int j = 1;

while(i < n){

    while(j < n){

        j \*= 2; // j = j \* 2

    }

    i++;

}

* + Answer: O(n \* log(n))

int function(int n){

    if(n <= 1){

        return n;

    }

    return function(n/2) + function(n/2);

}

* + Answer: O(2^log(n))

void function(int n){

    if(n == 0){

        return;

    }

    function(n-1);

    function(n-1);

}

* + Answer: O(2^n)

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

    for(int j = 0; j < n; j++){

        for(int k = 0; k < 10000000; k++){

            cout << i << " " << j << endl;

        }

    }

}

* + Answer: O(n^2)

Sorting

* Be able to manually sort an array for all three sorts
  + Example:  
    Manually sort the following array using insertion sort: {5, 6, 8, 1, 9, 3}

**5**, 6, 8, 1, 9, 3  
**5, 6,** 8, 1, 9, 3

**5, 6, 8,** 1, 9, 3

**1, 5, 6, 8,** 9, 3

**1, 3, 5, 6, 8,** 9

**1, 3, 5, 6, 8, 9**

* + Try: Manually sorting by bubble and selection.
* Be able to identify what sort is occurring given the sorting trace.
  + Example:

34, 15, 19, 14, 23

14, 15, 19, 34, 23

14, 15, 19, 34, 23

14, 15, 19, 34, 23

14, 15, 19, 23, 34

Answer: Selection Sort

* Write the function that sorts an array with O(n^2) best time.

Recommendation: Look over TA presentations in the Lab section of Blackboard for more examples.