**Linked List Review**

A linked list is a dynamic, linear data structure used to store data in non-contiguous memory (unlike arrays, which *do* store data in contiguous memory)

1. Explain the differences between a Single, Doubly and Circular Linked List

* **Singly Linked List - each node is connected by one next pointer.**
* **Doubly Linked List - each node is connected with one previous pointer and one next pointer.**
* **Circular Linked List - the tail of the linked list points to the head**

1. What are some public members we can include in the Linked Lists’ Class? What about private members?

Private Members:

* Head pointer
* Tail pointer (optional)
* Size (optional)

Public Members:

* Constructors/Destructors
* Linked List Functions
  + Insert
  + Remove
  + Print

1. Why do we save the head and tail as private members?

We don’t want the user to access the head and tail. To keep track of the first and last node of the LinkedList.

1. What kind of memory allocation do we use with Linked Lists?

Dynamic memory allocation

1. What are the differences between Linked Lists and Arrays?
2. Name the advantages and disadvantages of using Linked Lists.

A diagram of a red and white box

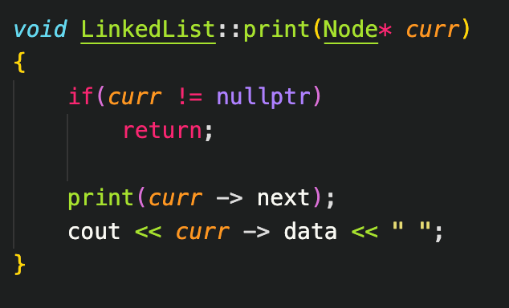
Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

1. Implement the addAtHead(), addAtEnd(), and addAtMiddle() functions.
2. Implement the removeAtHead(), removeAtEnd(), and removeAtMiddle() [Based on Value and Position] functions.
3. Name the difference between the 2 snippets of code:

Prints in reverse, prints in normal

A computer screen with colorful text

Description automatically generated

1. Implement a Linked List function that reverses a Linked List.

struct node{

int val;

node\* next;

};

node \*reverse(node \*head){

node\* curr = head;

node\* prev = nullptr;

node\* next = nullptr;

while(curr!=nullptr){

next=curr->next;

curr->next= prev;

prev=curr;

curr=next;

}

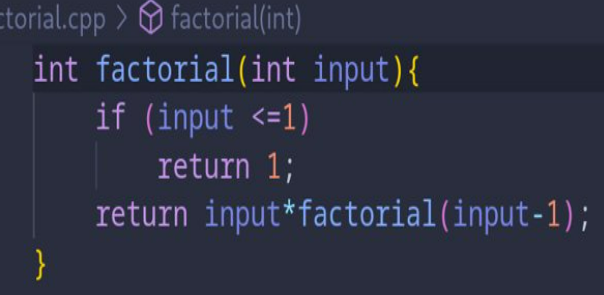
head=prev;

return head;

}

**Recursion Review**

1. Write a recursive function which returns the factorial of a given number.



1. Given a number n, print n-th Fibonacci number.

**int fibonacci (int x){**

**if(x<=1){**

**return n;**

**}**

**return fibonacci(n-1)\* fibonacci(n-2)**

**}**

1. Write a recursive function which given the number n, returns the sum of all numbers 0 to n.

#include <iostream>

using namespace std;

int ztoNum(int x){

if(x==0)

return 0;

return x + ztoNum(x-1);

}

int main() {

int num;

cin >> num;

cout << ztoNum(num) << endl;

return 0;

}

1. Given a string, write a recursive function that checks if the string is a palindrome.
2. Write a recursive function which prints every other node in a Linked List.

1. Write the output of the following recursive function.

A screen shot of a computer code

Description automatically generated

**Sorting Algorithms Review**

1. Write the Bubble Sort Algorithm Implementation
   1. Both v1 and v2
   2. What is the Time Complexity?
   3. Explain in words how the algorithm works
2. Write the Selection Sort Algorithm Implementation
   1. What is the Time Complexity?

A close up of black text

Description automatically generated

* 1. Explain in words how the algorithm works

1. Write the Insertion Sort Algorithm Implementation
   1. What is the Time Complexity

A black text on a white background

Description automatically generated

* 1. Explain in words how the algorithm works

1. Perform All Sorts on the Following Unsorted Array (Tracing)
   1. [12, 7, 5, 1, 3]

**Arrays Review**

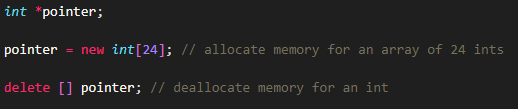
1. What are the main differences between statically and dynamically fixed arrays?
2. What do the “new” and “delete” keywords do?

* The operator ***new*** allocates memory (a variable) of the designated type and returns a pointer to it.

A black background with white text

Description automatically generated

* When a dynamic variable is no longer needed, it can be destroyed; that is, its memory can be deallocated. The C++ operator ***delete*** is used to destroy dynamic variables, so that its memory space can be allocated again when needed.



1. Name the differences between stack and heap memory. Which does static memory use? Which does dynamic memory use?
2. What are the advantages of using dynamic arrays over static ones?

**Time Complexity Review**

1. Order the time complexities from **fastest to slowest**.

O(n \* log(n)), O(n), O(n!), O(n^2), O(1), O(log(n)), O(2^n)

* O(1):  Constant time. Accessing array data, no matter the size of array. (Best)​
* O(log n): Logarithmic time. Divide and conquer methods, splitting arrays (Very Good)​
* O(n): Linear time. Time complexity grows proportionate to input size (Good)​
* O(nlogn): Linearithmic Time. Divide and Conquer methods of O(n) steps (Worst and Best)​
* O(n^2 ): Quadratic Time. Nested loops, traversing, 2D arrays, sorting (Not good)​
* O(2^n ): Exponential Time. Growth rates doubles with each addition to input (Very bad)​
* O(n!): Factorial Time. Example would be a Bogsort. (Extremely Bad) ​

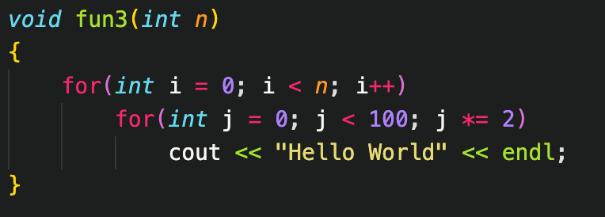
1. Review “Week 4 – Big O Notation” Slides for Time Complexity Examples
2. Name the Time Complexities to the following:

A black screen with white text

Description automatically generated A computer screen shot of a black screen with colorful text

Description automatically generated

A black background with colorful text

Description automatically generated 

A computer screen with colorful text

Description automatically generated 