**Data Structures and Algorithms**

**STUDENT MANAGEMENT SYSTEM  
Using Linked Lists**

**\_\_\_**



**INTRODUCTION**

This projects implements linked lists , using classes .

User Defined Header Files and functions

**Code Formatting**

Followed the , Google C++ styling Guide .

* **Include Files**

At the beginning of the file , in the order :

. system headers (stdio.h)

. Cpp standard headers (iostream)

. user defined headers

* **Naming Conventions**

Variable wont start with a number , cannot use the Cpp pre-defined keywords. User-defined class names and Functions are in the Pascal-Case ( Each letter is Capital without underscores ) .Variable names are in Lower-Case with underscore between words .

* **Punctuations**

Using the { } brackets the starting one will be on the same line as the statement

Closing ones are below the last line of the code .

Space between statement and the statement .

Space between the operators and the operands .

* **Indentation**

Two spaces at a time .

Line length utmost should be under 80 characters long .

Example Code Following the Above Rules ; 

**File Structure**



* **Driver.h**
* **LinkedList.h**
* **Node.h**
* **Others.h**
* **Main.cpp**

**Driver.h :**

This header files contains the code for two functions .

* Menu ( )

This function displays the main list of options that are basically the functions defined in the LinkedList.h file in the LinkedList class . It returns a int type variable that is then plugged to the second function in this header file .

* Executer ( )

This function takes multiple parameters :

* An int for the switch (choice)
* An object of the linked list class (obj)
* Three int for the loading animation ( time , length , code)

Switch case takes in the int and executes the case based on it . passes the linked list obj and rest of the parameters .

At the end after the break statement executes the code returns to the main.cpp

Where a goto function is acting as the game loop that keeps the code in recursion like scenario.

**LinkedList.h :**

Inside this list major work is done via the functions that are manipulation the data using the pointers . it has following functions :

* insert( )

Take in the parameters and maps them to variables in the temporary node that is created in heap and then mounted into the list , All by using the arrow (->) operator on pointers.

* AvailableData( )

This is a sub-function user cannot call it directly , but it displays data that can assist the inputs in other functions .

* Search ( )

This takes in a unique int , and runs through the linked list and compare this unique int to the Roll\_number variable that is in the node .

And then displays the data the is at that address .

* Count ( )

It traverse via the list and adds up to a variable .

* Update ( )

Similar to the Search( ) function in operation , it uses the Roll\_number to find a specific node .

This function also displays the output of AvailableData( ) function .

* Delete ( )

This takes in a unique int , and runs through the linked list and compare this unique int to the Roll\_number variable that is in the node . Then swaps that node to the one previous to basically terminating the link between node n+1 and n-1 and establishing a link between node n and n-1 .

* Show ( )

Similar to count at each one of its iteration it shows the data that is present at that node .

**Node.h :**

This class and the raw structure of the single link of the list . but the most basic component is the pointer of the Node class itself .

**Others.h :**

This class contains the code for the loading bar , it takes in 3 parameters via which we can control the following attributes of the loading bar :

* Length
* Speed
* Color

It uses the extended ASCII character set to display block character in a single line without spaces . Which creates the animation of the loading bar .

**Main.cpp :**

This is the main functions that has the declaration for the linkedlist object and it calls the executor function .It also passes the three variables for the loading bar .