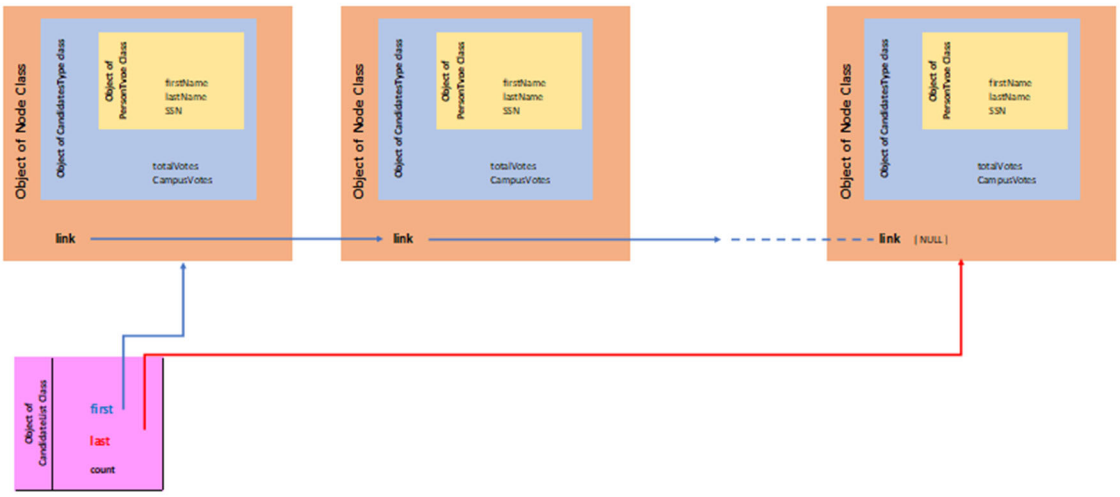
Final Project – Part c

For this part, you will need to complete the class CandidateList that creates a singly-linked list of nodes containing objects of the class CandidateType and a Pointer to the next node.

The CandidateList interface will have a class Node (implementation will be provided) that creates nodes storing a CandidateType object and a pointer link that points to the net node. The CandidateList class will contain a pointer that points to the first node in the list, and will contain a pointer that points to the last node of the list. CandidateList class will also contain an int count to keep track of the number of nodes in the list. Below is a visual illustration of the CandidateList class.



CandidateList Class	
Member variables	A pointer named **first** that points to the first node. A pointer named **last** that points to the last node. An integer variable named **count** that stores the number of nodes in the list.
Default Constructor	Initializes all member variables.
Function **addCandidate**	**Parameters**: An object of the CandidateType class. Inserts nodes to the **back** of the list. You have a pointer pointing to the back of the list; therefore, there is NO need to traverse the list.
Function **getWinner**	Traverses the list to find the candidate who has the highest numbe of votes, and returns the social security number assoicated with that candidate. If the list is empty, output the error message: "=> List is empty" and return 0;
Function **searchCandidate**	**Parameters**: A social security number. Tranverses the list to find the candidate with the given social security number and returns true if the candidate is found and false otherwise.

	Use a while loop so that you can stop when the candidate is found -> You are not allowed to use "break" or "continue". If the list is empty, output the error message "=> List is empty". If the candidate was not found, output the error message: "=> SSN not in the list".
Function **printCandidteName**	**Parameters**: A social security number. Traverse the list to find the candidate with the given social security number and prints out the name using the **printName** funcion of the **PersonType** class. Use a while loop so that you can **stop** the loop when the candidate is found **->** you are **NOT** allowed to use "break" or "continue". If the list is empty, output the error message "=>l List is empty." If the candidate was not found, output the error message "=> SSN not in the list."
Function **printAllCandidates**	Traverse the list to print all candidates using the **printCandidateInfo** function of the **CandidatetType** class. If the list is empty, output the error message "=> List is empty."
Function **printCandidateCampusVotes**	**Parameters**: A social security number and a division number. Prints out all the division votes for a given candidate, using the getVotesByCampus function of the CandidateType class. Use a while loop so that you can stop the loop when the candidate is found -> You are **NOT** allowed to use "break" or "continue" If the list is empty, output the error message "=> List is empty."
Function **printCandidateTotalVotes**	**Parameters**: A social security number Traverses the list to find the candidate with the given social security number and prints out the total number of votes using the getTotalVotes function of the CandidateType class. Use a while loop so that you can stop the loop when the candidate is found -> You are **NOT** allowed to use "break" or "continue" If the list is empty, output the error message "=> List is empty."
Function **destroyList**	Traverse the list to delete each node and reset all member variables to their default value.
Destructor	Calls the function destroyList.

Create a Main.CPP file,

Create a **Candidate_data.txt** file that contains some testing data to test your program.

file should be placed in the Resource Files folder of your project. The file contains a list of candidate to add to the linked list your program creates. Each line contains a social security number, a first name, a last name and four integers indicating the votes by campus(first integer for campus 0, second integer for campus 1, and so on):

123456789 Eddard Stark 89 41 45 5

The file ends with “-999” to stop the loop when reading the data

From Main.cpp call the appropriate functions to insert each candidate in a list.