Part A – Extend/Complete our class code
Task 1:
Finish writing the definitions of the member functions of the classes arrayListType and unorderedArrayListType that we did not do in class together. Also, write a program to test your function, add suitable makefile (compiler is clang++ please no make run scripts in your makefile, executable produced should may not have ".exe" file extension). Folder name: Task1

Task 2:
Extend our work:
Write the definition of the class template orderedArrayListType, derived from the class arrayListType, to implement an ordered list. add suitable makefile (compiler is clang++ please no make run scripts in your makefile). Folder name Task2

Part B (Sydney Stock eXchange)
Write a program to help my Sydney based stock trading company automate its systems. The company invests only in foreign commodities on the Sydney stock market SSX. At the end of each trading day, the company would like to generate and post the listing of its stocks so that investors can see how their holdings performed that day. We assume that the company invests in, say, 10 different stocks. The desired output is to produce two listings, one sorted by stock symbol and another sorted by percent gain from highest to lowest.

The input data is provided in a file in the following format: symbol openingPrice closingPrice todayHigh todayLow prevClose volume For example, the sample data is:

```
MSMT    112.50  115.75  116.50  111.75  113.50  6723823
CBA     67.50   75.50   78.75   67.50   65.75   378233
```

The first line indicates that the stock symbol is MSMT, today’s opening price was 112.50, the closing price was 115.75, today’s high price was 116.50, today’s low price was 111.75, yesterday’s closing price was 113.50, and the number of shares currently being held is 6723823.The listing sorted by stock symbols must be of the following form:

```
****************** Aussie Foreign Investor's Heaven **********
****************** Financial Report ***************************

Stock                   Today           Previous        Percent
Symbol  Open    Close   High    Low     Close           Gain    Volume
------  -----   -----   -----   -----   ------          ------  ------
```

ABC	123.45	130.95	132.00	125.00	120.50	8.67%	10000
AOLK	80.00	75.00	82.00	74.00	83.00	-9.64%	5000
CSCO	100.00	102.00	105.00	98.00	101.00	0.99%	25000
IBD	68.00	71.00	72.00	67.00	75.00	-5.33%	15000
MSET	120.00	140.00	145.00	140.00	115.00	21.74%	30920

Closing Assets: $9628300.00

Project in 3 steps. In the first step (part A), design and implement a newString object and a stock object.
In the second step (part B), design and implement an object to maintain a list of stocks.

Step A. (myString and Stock Objects)
Implement the newString object design defined in the myString api(.h) provided. Design and implement the stock object. Call the class that captures the various characteristics of a stock object stockType. The main components of a stock are the stock symbol (newString), stock price, and number of shares. Moreover, we need to output the opening price, closing price, high price, low price, previous price, and the percent gain/loss for the day. These are also all the characteristics of a stock. Therefore, the stock object should store all this information.
Perform the following operations on each stock object:

i. Set the stock information.
ii. Print the stock information.
iii. Show the different prices.
iv. Calculate and print the percent gain/loss.
v. Show the number of shares.

a.1. The natural ordering of the stock list is by stock symbol. Overload the relational operators to compare two stock objects by their symbols.

a.2. Overload the insertion operator, <<, for easy output.

a.3. Because the data is stored in a file, overload the stream extraction operator, >>, for easy input.

For example, suppose infile is an ifstream object and the input file was opened using the object infile. Further suppose that myStock is a stock object. Then, the statement:

```
infile >> myStock;
```

reads the data from the input file and stores it in the object myStock. (Note that this statement reads and stores the data in the relevant components of myStock.)

Step B.

Now that you have designed and implemented the class stockType to implement a stock object in a program, it is time to create a list of stock objects. Let us call the class to implement a list of stock objects stockListType. The class stockListType must be derived from the class listType (supplied code).
However, the class stockListType is a very specific class, designed to create a list of stock objects. Therefore, the class stockListType is no longer a template.

Add and/or overwrite the operations of the class listType to implement the necessary operations on a stock list.
The following statement derives the class stockListType from the class listType.

```
class stockListType: public listType<stockType>
{
        member list
};
```

The member variables to hold the list elements, the length of the list, and the maximum size of the list were declared as protected in the class listType. Therefore, these members can be directly accessed in the class stockListType.

Because the company also requires you to produce the list ordered by the percent gain/loss, you need to sort the stock list by this component. However, you are not to physically sort the list by the component percent gain/loss. Instead, you will provide a logical ordering with respect to this component.

To do so, add a member variable, an array, to hold the indices of the stock list ordered by the component percent gain/loss. Call this array sortIndicesGainLoss. When printing the list ordered by the component percent gain/loss, use the array sortIndicesGainLoss to print the list. The elements of the array sortIndicesGainLoss will tell which component of the stock list to print next.

Step C – final
Write a program that uses these two classes to automate the company’s analysis of stock data. All prices should be listed as AU$, cent fractions are not needed that is AU$ should show 2 decimal places. Add a suitable makefile(compiler is clang++ please no make run scripts in your makefile, executable produced should may not have ".exe" file extension). Folder name: SSX

Final note, this assignment grading is strictly by demonstration of working code, non runners incur a 50% penalty and would still need to demo and discuss their submissions with me. Appointment times (10/15 minute slots) will be announced in due course.

Supplied Code:

```
//Header file listType.h
#ifndef H_listType
#define H_listType

#include <iostream>
#include <fstream>

using namespace std;

template <class T>
class listType
{
public:
    bool isEmptyList() const;
    // Function returns a nonzero value (TRUE)if list is empty,
    // otherwise it returns the value 0 (False)
    bool isFullList() const;
    // Function returns a nonzero value (TRUE)if list is full,
    // otherwise it returns the value 0 (False)
    void setLength(int len);
    int showLength() const;
    void search(T searchItem) const;
    // Search the list for searchItem
    // Postcondition: found is set to a nonzero value (TRUE)if
    //    searchItem is found in the list,
    //    otherwise found is set to 0(False)
    void insert(T newElement);
    // Inserte newElement in the list
    // Prior to insertion list must not be full
    // Postcondition: list is old list plus the newElement
    void deleteItem(T deleteElement);
    // if deleteElement is found in the list it is deleted
    // If list is empty output the message "Cannot delete from the
    // empty list"
    // Postcondition: list is old list minus the deleteItem if
    //   deleteItem is found in the list
    void sort();
    // sort the list
    // Precondition: list must exist
    // Postcondition: list elements are in ascending order
    void print() const;
    // Output the elements of the list
    void getList(ifstream&);
    // read and store elements in the list
    // Postcondition: length = number of elements in the list
    //    elements = array holding the input data
    void destroyList();
    // Postcondition: length = 0
```

```
    void printList() const;
    // Output the elements of the list
    listType(int listSize);
    // constructor with parameters
    // Create an array of size specified by the parameter listSize
    //  Postcondition: elements contains the base address
    //    of the array, length = 0 and maxsize = listSize
    listType();
    // default constructor
    // Create an array of 50 components
    //  Postcondition: elements contains the base address
    //    of the array, length = 0 and maxsize = 50
    ~listType();
    // destructor
    // delete all elements of the list
    // Postcondition: array elements is deleted
protected:
    void binarySeacrh(T searchItem,
                      int& found, int& index);

    int maxSize; // maximum number that can be stored in the list
    int length;  // number of elements in the list
    T *elements; //pointer to the array that holds list elements
};


// constructor to set the array size specified by the user
template <class T>
listType<T>::listType(int listSize)
{
    maxSize = listSize;
    length = 0;
    elements = new T[maxSize];
}

template <class T>
listType<T>::listType()  // default constructor
{
    maxSize = 50;
    length = 0;
    elements = new T[50];
}

template <class T>
listType<T>::~listType()  //destructor
{
    delete [] elements;
}
```

```
template <class T>
bool listType<T>::isEmptyList() const
{
    return (length == 0);
}

template <class T>
bool listType<T>::isFullList() const
{
    return (length == maxSize );
}

template <class T>
void listType<T>::sort()   //selection sort
{
    int i, j;
    int min;
    T temp;

    for (i = 0; i <length; i++)
    {
        min = i;
        for (j = i+1; j < length; ++j)
            if (elements[j] < elements[min])
                min = j;

        temp = elements[i];
        elements[i] = elements[min];
        elements[min] = temp;
    }//end for
}//end sort

template <class T>
void listType<T>::print() const
{
    int i;

    for (i = 0; i < length; i++)
        cout << elements[i] << endl;
    cout << endl;
}//end print

template <class T>
void listType<T>::getList(ifstream& infile)
{
    int i;

    for (i = 0; i < length; i++)
        infile >> elements[i];
}
```

```
template <class T>
void listType<T>::search(T searchItem) const
{
    int found;
    int index;

    binarySeacrh(searchItem,found,index);

    if (found)
        cout << "Item is in the list" << endl;
    else
        cout << "Item is not in the list" << endl;
}

template <class T>
void listType<T>::binarySeacrh(T searchItem,
                               int& found, int& index)
{
    int first = 0;
    int last = length -1;
    int mid;

    found = 0;

    while( !found && (first <= last))
    {
        mid = (first + last) / 2;

        if (elements[mid] == searchItem)
            found = 0;
        else if (elements[mid] > searchItem)
            last = mid - 1;
        else
            first = mid + 1;
    }

    this->loc = mid;
}

template <class T>
void listType<T>::setLength(int len)
{
    length = len;
}

template <class T>
int listType<T>::showLength() const
{
    return length;
```

```
}

#endif
```