LAB 6

CSE225L

Sorted List (Array—Based)

In this lab, we will:

- Design and implement the List ADT, where the items are sorted.
- Implement the List ADT using an array—based structure.
- Create the **SortedType** class with methods for insertion, searching, deletion, and resetting the list.
- Test the functionality of the **SortedType** class by performing various operations.
- Create a **timeStamp** class to represent the time of day and perform operations on a list of **timeStamp** objects.

SORTED LIST (ARRAY-BASED)

sortedtype.h

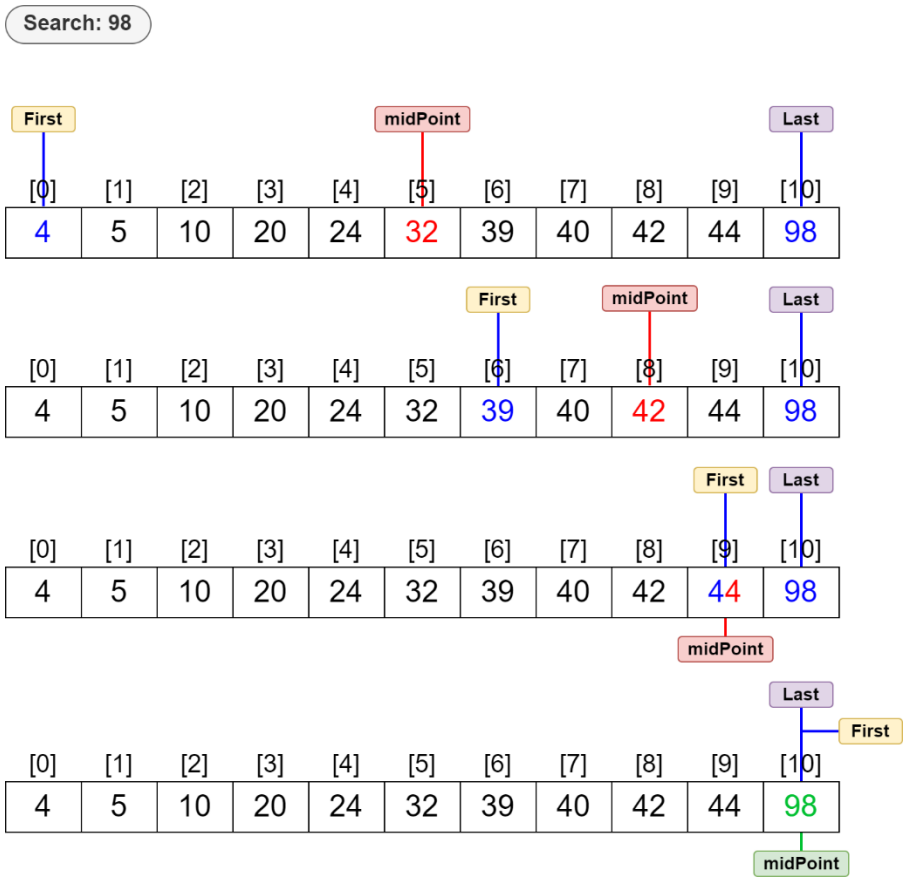
```
#ifndef SORTEDTYPE_H
#define SORTEDTYPE_H

const int SIZE = 5;

template <class T>
class SortedType
{
private:
    T *data;
    int currentSize;
    int pointTo;

public:
    SortedType();
    ~SortedType();
    int Length();
    bool IsFull();
    void MakeEmpty();
    void Insert(T value);
    void Search(T value, bool &found);
    void Delete(T value);
    void GetNext(T &value);
    void Reset();
};

#endif // SORTEDTYPE_H
```

sortedtype.cpp

```
#include "sortedtype.h"
#include <iostream>
using namespace std;

template <class T>
SortedType<T>::SortedType()
{
    data = new T[SIZE];
    currentSize = 0;
    pointTo = -1;
}

template <class T>
SortedType<T>::~SortedType()
{
    delete[] data;
}

template <class T>
int SortedType<T>::Length()
{
    return currentSize;
}
```

```cpp
template <class T>
bool SortedType<T>::IsFull()
{
    return (SIZE == currentSize);
}

template <class T>
void SortedType<T>::MakeEmpty()
{
    currentSize = 0;
}

template <class T>
void SortedType<T>::Insert(T value)
{
    if (IsFull())
    {
        cout << "Error: List is full" << endl;
    }
    else
    {
        int i = 0;

        while(i < currentSize)
        {
            if (value > data[i])
            {
                i++;
            }
            else
            {
                for (int j = currentSize; j > i; j--)
                {
                    data[j] = data[j - 1];
                }
                break;
            }
        }
        data[i] = value;
        currentSize++;
    }
}
```

```
template <class T>
void SortedType<T>::Search(T value, bool &found)
{
    int midPoint;
    int first = 0;
    int last = currentSize - 1;
    found = false;

    while(first <= last)
    {
        midPoint = (first + last) / 2;

        if(value < data[midPoint])
        {
            last = midPoint - 1;
        }
        else if (value > data[midPoint])
        {
            first = midPoint + 1;
        }
        else
        {
            found = true;
            value = data[midPoint];
            break;
        }
    }
}
```



```
template <class T>
void SortedType<T>::Delete(T value)
{
    bool found = false;
    int i = 0;

    while (i < currentSize)
    {
        if (data[i] == value)
        {
            found = true;
            break;
        }
        else
        {
            i++;
        }
    }

    if (found)
    {
        while (i < currentSize)
        {
            data[i] = data[i + 1];
            i++;
        }
        currentSize--;
    }
    else
    {
        cout << "Error: Item could not be found in the list" << endl;
    }
}

template <class T>
void SortedType<T>::GetNext(T &value)
{
    pointTo++;
    value = data[pointTo];
}

template <class T>
void SortedType<T>::Reset()
{
    pointTo = -1;
}
```

SORTED LIST (ARRAY-BASED)

TASKS:

Instructions:

- Create the driver file (main.cpp) and perform the following tasks.
- You cannot make any changes to the header (**.h**) or source (**.cpp**) files of the **SortedType** class.

Operation	Input Values	Expected Output
Create a list of integers		
Insert four items	5 4 2 1	
Print the list		1 2 4 5
Insert one item	7	
Insert one more item	12	Error: List is full
Print the list		1 2 4 5 7
Search 6 and print whether found or not		Item is not found
Search 5 and print whether found or not		Item is found
Print if the list is full or not		List is full
Delete 1		
Print the list		2 4 5 7
Delete 4		
Print the list		2 5 7
Delete 16		Error: Item could not be found in the list
Write a class **timeStamp** that represents a time of the day. It must have variables to store the number of **seconds**, **minutes** and **hours** passed. It also must have a function to print all the values. *You will also need to overload a few operators.*		
Create a list of objects of class **timeStamp**.		
Insert 5 'time' values in the format ssmmhh	15 34 23 13 13 02 43 45 12 25 36 17 52 02 20	
Delete the **timeStamp** 25 36 17		
Print the list		13: 13: 02 43: 45: 12 52: 02: 20 15: 34: 23