North South University
Department of Electrical and Computer Engineering
Semester: Summer 2025 | Section: 11
CSE225L (Data Structures and Algorithms)
Midterm Exam
Time: 60 Minutes | Marks: 20

ANSWER ALL QUESTIONS

Question 1 [10 Marks | CO4]

You are given a partial implementation of a templated `SortedType` class, which maintains a sorted list of items of any data type. The `Search` function is currently unimplemented.

Task A:

Implement the `Search` function using the **Binary Search algorithm** in the `SortedType` class. Your implementation should correctly identify whether an item exists in the list and set the `found` flag accordingly.

Task B:

Using your NSU ID (e.g., `191211404`) as individual digits, complete the following operations:

Operation	Input Values	Expected Output
Create a list of integers		
Insert your NSU ID digits	1 9 1 2 1 1 4 0 4	
Print the list		0 1 1 1 1 2 4 4 9
Search for an existing digit	4	Found
Search for a non-existing digit	8	Not Found

Question 2 [10 Marks | CO4]

Write a class named `Movie` that represents a movie record. The class must have:

- Member variables to store the movie's **title**, **director**, and **release year**.
- A method to **print** all the values in a formatted manner.
- If needed, overload appropriate operators for sorting or displaying.

Task:

Perform the following operations using your `Movie` class:

Operation	Input Values	Expected Output
Create a list of `Movie` objects		
Insert 3 movie records	Inception, Nolan, 2010Matrix, Wachowskis, 1999Interstellar, Nolan, 2014	
Print the list		Matrix, Wachowskis, 1999Inception, Nolan, 2010Interstellar, Nolan, 2014

```
// 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;
}

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)
{
  // Start writing your code from here
}

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;
}
```