University of Central Punjab

**Faculty of Information Technology**

# Data Structures and Algorithms

# Spring 2021

|  |  |  |
| --- | --- | --- |
| **Lab 01** | |  |
| **Topic** | * Simple sorting algorithms — Selection sort * Simple searching algorithms — Linear search, binary search * Working with classes and multiple files. |
| **Objective** | The basic purpose of this lab is to revise some preliminary concepts of C++ that has been covered in the course of Introduction to Computing and Programming Fundamentals and Object Oriented Programming. |
|  | | |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* Submit only .CPP and .H files in a single ZIP folder.
* The name of each CPP file should be the question number. E.g., Q1.cpp
* The name of the ZIP folder should be your complete registration number. E.g., L1S21BSCS0000
* **void main() is not allowed. Use int main()**
* **You have to work in multiple files. i.e separate .h and .cpp files**
* **You are not allowed to use system**("**pause**")
* **You are not allowed to use any built-in functions**
* **You are required to follow the naming conventions as follow:**
  + **Variables:** firstName; (no underscores allowed)
  + **Function:** getName(); (no underscores allowed)
  + **ClassName:** BankAccount (no underscores allowed)

**Students are required to complete the following tasks in lab timings.**

# Selection Sort

Selection sort is a sorting algorithm, in which we repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array. Assume that we wish to sort the array in increasing (ascending) order, i.e. the smallest element at the beginning of the array and the largest element at the end. We begin by selecting the smallest element and moving it to the lowest index position. We can do this by swapping the element at the lowest index and the smallest element. We then reduce the effective size of the array by one element and repeat the process on the greater (sub)array. The process stops when the effective size of the array becomes 1 (an array of 1 element is already sorted).

## Pseudo code

Input: An unsorted *array, A of N elements*

Output: Sorted array, A

For I = 0:N-1

SmallSub = I

For J = I+1:N-1

If A[J] < A[SmallSub]

SmallSub = J

End-If

End-For

Swap ( A[I], A[SmallSub] )

End-For

# Linear Search

In computer science, linear search or sequential search is a method for finding a particular value in a list that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found.

## Pseudo code

Input: An unsorted *array,* *A of N elements* and *value* to be searched

Output: Index of searched element or -1 if not found

For I = 0:N-1

If ( A[I] == Value )

Return [I];

End-if

End-For  
return -1;

# Binary Search

Binary search is another searching algorithm, used to search a specific value (or index of value) from the *sorted array*.

In binary search, we first compare the *value to be searched* with the item in the middle position of the array. If there's a match, we can return immediately. If the key is less than the middle key, then the item sought must lie in the lower half of the array; if it's greater than the item sought must lie in the upper half of the array. So we repeat the procedure on the lower (or upper) half of the array.

## Pseudo Code

Input: An *Sorted* *array,* *A of N elements* and *value* to be searched

Output: Index of searched element or -1 if not found

low = 0, high = N-1;

while (low <= high)

mid = (low + high)/2;

If ( A[mid] == Value )

Return mid;

Else-if ( A[mid] < Value )

low = mid + 1;

Else

high = mid – 1;

End-if

End-For  
return -1;

## Task 1

Implement the Selection Sort Algorithm with the help of pseudo code given above.

## Task 2

Create a C++ class named **Student**, with the following private attributes:

1. regNo: char\*
2. CGPA: double

Your task is to instantiate array of 10 Students. Initialize all objects of the array with different values and perform the following Search algorithms on them on the basis of CGPA.

1. **Linear Search**
2. **Binary Search**

**These functions MUST be implemented as PUBLIC member functions of the class Student with the help of pseudo code given above.**

**NOTE:** In orderto perform Binary Search, the array must be sorted. So you can your selection sort function implemented in task for this purpose.

## Task 3

Define a class called Cake that has member variables to track the type of cake (either chocolate cake, sponge cake, or character cake) along with the size (either one tier, two tier, or three tier), the number of chocolate pieces on top and if there is a decoration of flowers added on top or not. Add appropriate Constructor(s) and Destructor for initialization. Include mutator (setter) and accessor (getter) functions for your class.

Create a **member function** that outputs a textual description of the cake object, like (*This cake is two tier, chocolate cake with flower decor and has 8 pieces of chocolates*). Also include a **member function** that computes the cost of the cake and returns it as a double (data type) according to the following rules:

One tier cake (any type) = $15 + $7 for flower decor + $0.5 \* no. of chocolate pieces

Two tier cake (any type) = $25 + $13 for flower decor + $1.0 \* no. of chocolate pieces

Three tier (any type) = $35 + $19 for flower decor + $1.5 \* no. of chocolate pieces

Write a test program that demonstrates capabilities of the class.

## Task 4

Create a C++ generic abstract class named as **List**, with the following:

**Attributes:**

1. Type \* arr;
2. int maxSize;
3. int currentSize;

**Functions:**

virtual void addElement(Type) = 0;

* Should add the element at the last position of the **List**

virtual Type removeElement() = 0;

* Should remove the element from the last position of the **List**
* Write parameterized constructor with default arguments for the above class.
* Write Copy constructor for the above class.
* Write Destructor for the above class.

Using the class above, make a class named as **MyList**, having following additional functionalities:

**bool** [**empty()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns whether the MyList is empty or not

**bool** [**full()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) **:** Returns whether the MyList is full or not  
**int** [**size()**](https://www.geeksforgeeks.org/stack-empty-and-stack-size-in-c-stl/) : Returns the current size of the MyList   
**Type** [**last ()**](https://www.geeksforgeeks.org/stack-top-c-stl/) : Returns the last element of the MyList

* Write parameterized constructor with default arguments for the above class.
* Write Copy constructor for the above class.
* Write Assignment operator for the above class.
* Write indexing/subscript operator for the above class
* Write Destructor for the above class.

Now write a global function show list which should display all the contents of the list.

void showList(MyList <Type> s);

**Hint:** Use indexing operator here

Instantiate several objects of MyList, test all the functions of MyList on them and then display them through showList function.