AA-Homework 2

Question 1

1) Write BubbleSort Implementation by following the below steps.
    a) Write the code for **BubbleUp**, and after that make a function BubbleSort, first make a for-loop which runs N-1 times BubbleUp and test whether the array is sorted or not?
    b) Then modify the **BubbleSort** calls so that it does not compare with the already placed maximas already reached at the end, by shrinking Size parameters.
    c) Now change the **BubbleUp** implementation so that it becomes a boolean returning function and BubbleSort as one line function such that it keeps calling **BubbleUp** until no swap happens.
2) Given examples of Complexity of Best, Worst Case analysis of the above three parts.
3) Is BubbleSort Stable? A Sort is called stable if there are two values in the array which are equal that the value which comes first should come first after sorting.

- **MAKE SURE YOU GIVE TWO IMPLEMENTATIONS OF THE BUBBLESORT first integers arrays and than by using vector<int> implementations (as we did in the class). Without vector implementation you will lose 15 points.**

Question 2

1. **SelectionSort** Implementation using the following in mind?
    a. Write the **SelectMinIndexWithinRange** function as described and discussed in class.
    b. Analyse its Time Complexity analysis
    c. **SelectionSort** using **SelectMinIndexWithinRange** function
2. Is **SelectionSort** Stable? A Sort is called stable if there are two values in the array which are equal that the value which comes first should come first after sorting.
3. Where SelectionSort beats BubbleSort? And where BubbleSorts beat SelectionSort?

- **MAKE SURE YOU GIVE TWO IMPLEMENTATIONS OF THE SELECTIONSORT first integers arrays and than by using vector<int> implementations (as we did in the class). Without vector implementation you will lose 15 points.**

Question 3

1. Write the code of **InsertInSortedSubarray.**
    i. **InsertInSortedSubarray(int A[ ] , int ii)**
    ii. **InsertInSortedSubarray(vector<int> & Vs , int ii)**
2. InsertionsSort using **InsertInSortedSubarray** method.
3. **Its Time Complexity Analysis**
4. Is **SelectionSort** Stable? A Sort is called stable if there are two values in the array which are equal that the value which comes first should come first after sorting.
5. Give its Best and Worst Time Cases.

- **MAKE SURE YOU GIVE TWO IMPLEMENTATIONS OF THE InsertionSort first integers arrays and than by using vector<int> implementations (as we did in the class). Without vector implementation you will lose 15 points.**

Question 4

A. Write a program that takes upto 2MB integers (Capacity) …….. with -99.

- Further your program should be able to identify the distinct elements and store it in an array named as **DistinctArray** and then using the print function it should display the **DistinctArray**. A distinct element of an array is the one which has occurred only once in an array.
  Drive its time complexity?

  vector<int> DistinctElementsArray(vector<int> & Vs)

- Further your program should be able to identify the unique element and store it in an array named as **UniqueArray** and then using the print function it should display the **UniqueArray**. Unique elements of an array are such that if an element appears more than once then it should be printed once only.
- Now your program should make sure that the DistinctArray should be sorted in increasing order.

  RECOMMENDATION FOR SORTING
  Use STL Sorting Algorithm (both for ascending and descending)

Sample Input :
20 11 12 20 16 15 12 16 8 12 -99
Sample Output :
Distinct Elements are: 11 15 8
Distinct Element in Sorted(Increasing order) are: 8 11 15
Unique Elements are: 20 11 12 16 15 8
Unique Element in Sorted(Decreasing order) are: 20 16 15 12 11 8

B. Write a program that takes upto 20 integers (Capacity) …….. with -99.
Further your program should be able to sort all the even indices of array in decreasing order.

Sample Input :
20 11 **12** 16 **16** 15 **13** 20 **8** 12 -99
Sample Output :
20 11 **16** 12 **13** 15 **12** 16 **8** 20

C Write a program that takes upto 20 integers (Capacity) ..... with -99.
Further your program should be able to sort all the elements on odd indexes of an array in increasing order.

Sample Input :
20 **11** 12 **14** 16 **15** 19 **13** 18 **17** -99
Sample Output :
20 **11** 12 **13** 16 **14** 19 **15** 18 **17**

3C. Write a program
Further your program that should merge 3A and 3B.

D. Write a program that takes upto 20 integers (Capacity) ..... with -99.
Further your program should be able to sort all the even elements of an array in decreasing order. Note that this problem is a different problem as for each number it has to first search the next odd element and then compare and swap (if the condition for not sorted is fulfilled).

Sample Input :
20 11 **12** **14** **16** 15 19 13 **18** 17 -99
Sample Output :
12 11 **14** **16** **18** 13 15 17 **20** 19

2D. Write a program that takes upto 20 integers (Capacity) ..... with -99.
Further your program should be able to sort all the odd elements of an array in increasing order. Note that this problem is a different problem as for each number it has to first search the next odd element and then compare and swap (if the condition for not sorted is fulfilled).

Sample Input :
20 **11** 12 14 16 **15** **19** **13** 18 **17** -99
Sample Output :
20 **11** 12 14 16 **13** **15** **17** 18 **19**

4C. Write a program that takes upto 20 integers (Capacity) ..... with -99.

Question 5

- Implement the class **MyStack** which we implemented in the class.
  - Using Static/Fixed size implementation
  - **Using Dynamic Arrays(Vector) composition inside Stack class**

Applications: **Using Stack Solve the BalancedParenthesis problem.**

A. With only () types of parenthesis **5** **( ))) ((( )** **0 0 0 0 0 0 0)** **((((( )))))** **0 0 0 0 0 0 0)** **((((0 0 0 0 0 0 0)** **1. Using only maintaining count** **2. Using Stack**	**B. With all {} () [] types of parenthesis** **5** **{ ( ) } [ ]** **{ (0 } [ ]** **{ ( ) } [ ] { ( ) } [ ]** **{ (0 } [ ] { ( ) } [ ] { ( ) } [ ]** **{ (0 } [ ] { ( ) } [ ] { (0 } [ ] { ( ) } [ ] { ( ) } [ ]**

Question 6

A. How would you design a stack which, in addition to push, isempty, isFull and pop, also has a function min/max which returns the minimum element? Push, pop and min/max should all operate in O(1) time.

B. Write a program to sort a stack in ascending order. You should not make any assumptions about how the stack is implemented. The following are the only functions that should be used to write this program: **push | pop | peek | isEmpty.**

void Sort(stack<int> & S) // This must modify S such that on return S is sorted, You are only allowed to use another stack as extra memory only. HINT - NOT MORE THEN O(N^2)

C. Implement a **MyQueue** class which implements a queue using two stacks.

Question 7

A. Implement the class of Quack which is (Queue+Stack) at the same time i.e. it should have all the operations possible of Stacks and Queues, using only and only stacks as black boxes data structures (of STL). All operations must be AMORTIZED O(1) operations

CHALLENGE* (1 Page Write Up on this talk)

Arithmetic, Population and Energy - a talk by Al Bartlett

https://www.youtube.com/watch?v=O133ppiVnWY