CSE102: Data Structures and Algorithm
Assignment-2
Max Marks: 45

Instructions

- Use C++ programming language only.
- Code will be checked for plagiarism.
- Upload a Zip folder named <RollNo>_<YourName>. If not named appropriately, it will not be evaluated.
- For the coding questions, you must ensure that your code compiles and runs otherwise, these will be awarded **zero marks**.
- Using maps, sets and internal sorting functions is not allowed for given questions. If used will be awarded **zero marks**.
- Students might be required to give a demo of their codes.
- `<RollNo>_Q2.cpp` - Code for Q2 (Exclude object files)
- `<RollNo>_Q3.cpp` - Code for Q3 (Exclude object files)
- `<RollNo>_Q4.cpp` - Code for Q4 (Exclude object files)
- `<RollNo>_<YourName>.pdf` - Report including answers for Q1

---

1. (15 points) For the given three sorting algorithms: insertion sort, selection sort and merge sort select the most suitable sorting algorithm for the scenarios provided below and provide a rationale for your choice. Your justification carries more weight than your selection. Each sorting algorithm might be used more than once. If multiple sorting algorithms could fit a scenario, outline their advantages and disadvantages for each of them, and then pick the one that aligns best with the application. Clearly state and justify any assumptions you make and perform asymptotic analysis. "Best" should be evaluated based on **asymptotic running time**.

   (a) (5 marks) You have a custom data structure, $DS1$, that organizes k items and supports two operations:
   `DS1.get_at_index(j)` takes constant time, and
   `DS1.set_at_index(j, x)` takes $\Theta(k \log k)$ time.
   Your task is to sort the items in DS1 in-place.

   (b) (5 marks) Imagine you possess a fixed array A containing references to n comparable objects, where comparing pairs of objects takes $\Theta(\log n)$ time. Select the most suitable algorithm to sort the references in A in a manner that ensures the referenced objects appear in non-decreasing order.

(c) (5 marks) Suppose you are given a sorted array A containing n integers, each of which fits into a single machine word. Now, suppose someone performs some $\log \log n$ swaps between pairs of adjacent items in A so that A is no longer sorted. Choose an algorithm to best re-sort the integers in A.

2. (10 points) (a) (5 marks) Implement stack and queue from scratch using pointers in C++. They should have all the functions supported by stacks and queues, e.g., insert, pop, top, etc. (Plagiarism will be strictly checked.)

   (b) (5 marks) Use your stack implementation from above and solve the following problem:- Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return the next greater number for every element in nums. The next greater number of a number x is the first greater number to its traversing order next in the array, which means you could search circularly to find its next greater number.

```
Input: [1,2,3,4,3]
Output: [2,3,4,-1,4]
```

3. (10 points) Mitsuha is traveling to Tokyo to meet Taki Tachibana. She plans to prepare Itomori's best sushi for him, which requires three main ingredients: Bread (B), Salmon (S), and Corn (C). She expresses her favourite recipe as a string, such as "BBSCS", meaning Bread, Bread, Salmon, Corn, and Salmon.

   Mitsuha has n_b pieces of bread, n_s pieces of salmon, and n_c pieces of corn in her kitchen. The nearby shop offers all three ingredients (unlimited supply) at prices p_b yen for bread, p_s yen for salmon, and p_c yen for corn.

   With r yen to spend, what is the maximum number of sushi plates she can prepare, assuming she cannot break or slice any ingredients?

   Input:

   - Recipe: a string containing only letters B, S, C such that $len(string) \leq 100$
   - $1 \leq n_b, n_s, n_c \leq 100$
   - $1 \leq p_b, p_s, p_c \leq 100$
   - $1 \leq r \leq 10^{12}$

```
Example 1:
Recipe: BBBSSC
n_b = 6, n_s = 4, n_c = 1
p_b = 1, p_s = 2, p_c = 3
r = 4
Output: 2
```

```
Example 2:
Recipe: BSC
n_b = 1, n_s = 1, n_c = 1
p_b = 1, p_s = 1, p_c = 3
r = 1000000000000
Output: 200000000001
```

4. (10 points) You are playing Dota2 and are currently in the middle lane, which has n heroes. Each hero has the following attributes:

   - Position: Given as a 0-indexed integer array. All integers are unique. The index in the array refers to a particular hero, and the value at that index indicates the position.
   - Health: Given as a 0-indexed integer array. The index in the array refers to a particular hero, and the value at that index indicates the health.
   - Team: Given a string (R for radiant, D for Dire).

   All heroes move simultaneously at the same speed on the lane, each going in its given direction. Radiant moves up, and Dire moves down the middle lane. If two heroes ever meet in the same position, they have a battle. When this happens, the hero with less health is removed from the lane, and the health of the other hero drops by one. If both heroes have the same health, they both get removed.

   Return an array containing the health of the remaining heroes (in the order they were given in the input), after no further battle can occur.

   Constraint :

   - $1 \leq positions.length, healths.length \leq 10^5$
   - $1 \leq healths[i], positions[i] \leq 10^9$

```
Input: positions = [5,4,3,2,1], healths = [2,17,9,15,10], Team = "RRRRR"
Output: [2,17,9,15,10]
Explanation: No fight occurs in this example since all heroes move in the
same direction.
So, all the heroes' health is returned in initial order [2, 17, 9, 15, 10].

Input: positions = [3,5,2,6], healths = [10,10,15,12], Team = "RDRD"
Output: [14]
Explanation: There are 2 battles in this example. Firstly, hero 0 and hero 1
will fight, and since both have the same health, they will be removed.
Next, Hero 2 and Hero 3 will fight, and since Hero 3's health is less
it gets removed, and Hero 3's health becomes 15 - 1 = 14.
Only Hero 3 remains, so we return [14].
```

```
Input: positions = [1,2,5,6], healths = [10,10,11,11], Team = "RDRD"
Output: []
Explanation: hero 0 and Hero 1 will fight, and since they
have the same health, they are removed.
Hero 3 and 4 will fight, and since they have the same health, they are removed.
So, we return an empty array [].


Extra test cases :

Input: positions =  [2,19,46], healths = [42,45,2], Team =  "DRD"
Output: [42,44]

Input: positions =  [2,3,21,22,24,25,38,31], healths =[24,33,31,37,19,16,11,50],
Team =  "DDRDDRDR"
Output: [24,33,36,19,16,49]
```