**Indian Institute of Technology, Mandi**
**August - November 2019**

# CS202 - Data Structures and Algorithms

# Programming Assignment 1

Course Instructor : Aditya Nigam

Assignment created by : Ranjeet Ranjan Jha

14 September 2019

## Instructions

- Plagiarism is strictly prohibited. In case of violation, a zero will be awarded for this assignment as a warning and a quick F grade if repeated later.
- The questions starting with (*), are only for practise purpose. Hence, no need to submit these type of questions.
- Submit the complete code as cs202assignment1.zip file with a Makefile.
- For Problems, make different cpp files for each problem. For example, prob1.cpp.
- For Problems, Makefile should contain targets for each problem. For example, make prob1 should compile and generate executable of prob1.cpp.
- You can lookup this link on how to make makefile : `https://stackoverflow.com/questions/2481269/how-to-make-a-simple-c-makefile`.
- You must not use STL library classes and functions.
- You must only use seqLinearList.hpp wherever you require an array.
- For Sorting Algorithms and Problems, You will have to submit a report in which following things must be covered: Pseudocode of Algorithm, Time and Space Complexity for all cases (best, average, worst), Remarks (for sorting algorithms, on how it's different from other sorting algorithms, when should it be used and your personal insights).
- The deadline for submission is **Sunday, 22$^{th}$ September, 2019, 1200 HRS (12:00 noon)**. No late submissions will be entertained.
- Contact Ranjeet Ranjan Jha (7066896525) or Daksh Thapar (9592563214) for any queries.

# 1 Implement Sorting Algorithms

Implement the following sorting algorithms using C++ programming language and sort the input sequence in ascending order.

1. Insertion Sort
2. (*) Selection Sort
3. (*) Rank Sort
4. (*) Bubble Sort
5. Merge Sort
6. Quick Sort
7. Heap Sort

**Note:**

1. Program should take n inputs given from the user and display the sorted sequence.
2. Each array is implemented using the sequential linear list data structure **(seqLinearList.hpp)**.
3. Everyone should use the **seqLinearList.hpp** and **sorting.hpp** classes provided. Do not change the class names. It is expected to strictly use the interfaces provided in the classes to implement the tasks.

# 2 Problems

1. (*) Ross and Rachel are trying to find a new place to live together as a couple. They are in a dilemma. Their friends live in different parts of the city and they want to live close to all of them. All the houses in the city lie on the lattice(integral) points on a horizontal line (x-axis). You are given the x-coordinates of all the houses along that line in an array A of $n$ integers $A_1, A_2, ..., A_n$ . Your task is to find the sum of all the distinct x that will minimize the total of the distances between their (Ross and Rachel) and each friend's house. Note that they can also live at their friends' place if required. Print the answer modulo $10^9 + 7$.
2. There is a small primary school in town Gachibowli. There studies a girl named Pratibha. She is very talented and hard-working. Whenever she doesn't find any work she starts getting depressed. Today she has completed all her school homework and now she has nothing to do. To avoid depression she usually takes an array of length $n$ and starts playing with it. She first rolls a dice and according to a number she gets on dice she performs an operation on the array which is usually swapping some elements in the array.

If the number on the dice is even she divides it by two and gets the operation number to be performed. If the number on the dice is odd then she divides the number by two and adds one to it to get the operation number to be performed. Apart from this she also has an integer "head".

**Operation 1:** She takes the (head+1)th element(say it x) from the array and inserts it between the 1st and (head)th element of the array. She inserts it at $i^{th}$ place such that (i+1)th element is greater than x and (i-1)th element is smaller than x(if there are multiple such places then choose min "i"). If she doesn't find both the conditions true at the same time she puts it at a place where at least on of the two conditions holds. After that she increase the head by $1$.

**Operation 2:** She takes the minimum element between (head+1)th and (n)th index of the array and swaps it with (head+1)th element of the array. After that she increase the head by $1$.

**Operation 3:** She iterates from (head+1)th element till end of the array and swaps two elements if $a[i] > a[i+1]$.

She has done some k operations on the array and now it's time for her to go for her dance class. She wants you to validate if she has done any mistake while doing operations or not. (You should not use any other additional array other than array you are performing operations on.)

3. "Hardest choices require strongest wills (and so does this problem :-) )".

   Thanos is the sole survivor of his planet (Moon of Saturn) Titan who has a different perception of the world. According to him, the resources of universe are limited and the days are not too far when there would only be starving and death when the resources get exhausted. So he is on his own mission to save the universe by killing half population of every species in the universe. On his conquest, he had come across infinity stones and came to know that if he could harness the power of these relics then he could achieve his goal just by a snap of his fingers. So he compelled Eitri of Nidavellir, the weapon forger, to make him a gauntlet so as to harness the power of infinity stones. Now after many hardships, he finally had all infinity stones and then snaps his finger. The gauntlet was designed in a way that doesn't hinder the natural process of evolution by natural selection(to some extent). So it firstly divides the population into half and then exchanges at most $K$ healthy individuals from one half to the other in order to maximize the difference between the health metrics of both groups and kills the half which is lesser healthy. What is the algorithm that gauntlet applies? Implement it.

   **Solve it as efficiently as much you can (in terms of time and space complexity).**

   **Input:** An array A with values as health metrics of population and integer $K$.

   **Output:** Two arrays P and Q with Q being the population that has been killed.

4. (*) In the primary school of Gachibowli, every morning the students assemble in the school playground for prayer. There are 'N' lines formed at the time of prayer and each line is represented using an uppercase English alphabet. Students enter in the playground in groups of 'M'. Each student has an RollNo assigned by the school. Roll No of students also contains information about the line in which student is going to stand. RollNo's are assigned in such a way that students of same class stand in same line. Roll No is of the form XZ(ex $C123$) where X is an uppercase character denoting the line in which student has to stand and Z is a unique integer id assigned to each student. Students have returned after a long vacation. Today, the student are forming lines in a particular fashion. They want to stand behind the first friend they encounter in their line. Two students are friends if they have two common factors other than 1 in their id's(Ex: if the id's are 6,12 then they have 2,3 as factors and they are friends. Also 4,8 are friends as they have 2,2 as common factors but 3,6 are not friends). There are total of 'K' number of student in the school. Now the PTI Chaubey comes in and finds that the students are standing with their friends making noise. So he decided to arrange them into line according to their height. He will swap $2$ students in a line at a time and it will cost him on an average of $0.5$ sec for each swap. He wants you to help him minimize the time taken to rearrange students in all lines so that the prayer will start as soon as possible. Calculate the minimum time required to rearrange each of the $n$ lines.

   **Input:**

   $N$ : No of lines in playground($0 <= N <= 26$)

   $M$ : No of students in groups entering into ground.

   $K$ : Total number of students in school(It need not be multiple of $M$. In that case last group entering the ground will have ($KmodM$) no of students.)

   The next $k$ lines will contain two integers each corresponding to student id(integer) and his height(double)

   Then there will be ceil($K/N$) no of lines telling the students id's in each group entering the ground.

   **Output:** Output 'N' numbers showing minimum time required to rearrange each of the n lines(numbers should be in order of line index 'A' to 'Z'). (Use Linked List for mimicking each line in ground. Your program should be modular and clean. Try to solve it as efficiently as possible.)

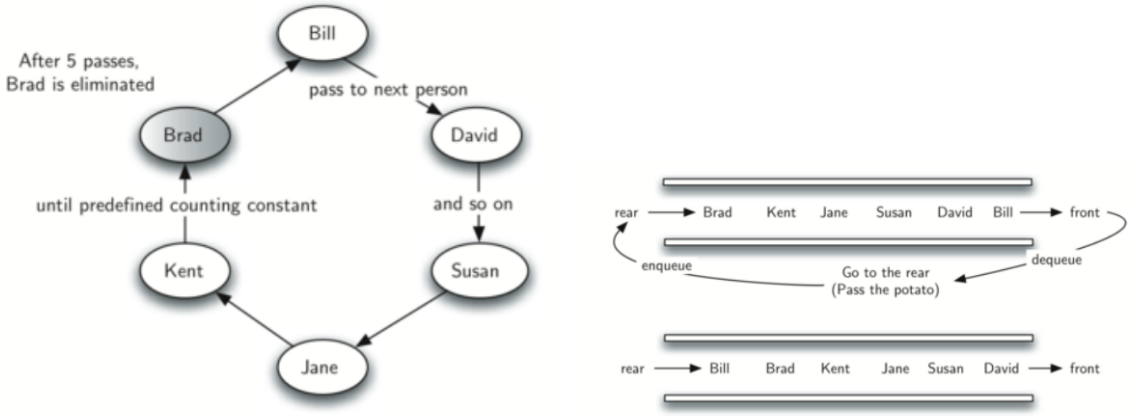
5. **Hot Potato (Josephus problem):**

   In this game children line up in a circle and pass an item from neighbor to neighbor as fast as they can. At a certain point (say after a round) in the game, the action is stopped and the child who has the item (the potato) is removed from the circle. Play continues until only one child is left.

   (a) **Players :** Let there are $n$ children.

(b) **Elimination Rule :** Every $i^{th}$ child will be removed after a round .

Devise a winning strategy, it mean that one can choose a strategy to be the last person left who will be declared as the winner.




(a) Six Children Playing Hot Potato (b) Queue Implementation

Figure 1: Hot Potato Setup

**Task Description**

You are required to write a $C++$ program which take the number of children ($n$) from user and output the winning strategy.

**Input Data and Format**

- Enter number of children : $n$
- Elimination Rule : $i$

**Expected Output for Correct and Incorrect Inputs**

Return: winning strategy i.e. a safe position s for some given values of $n$ and $i$.

**Example :** If n = 7 and i = 3, then the safe position is s = 4. The persons at positions 3, 6, 2, 7, 5, 1 are removed in order, and person at position 4 survives.

**Sample Output :**

$ hpotato

Please enter values of $n$ and $i$ : 5 2

The removal sequence is as follows -

[1] Firstly, the person at position 2 is removed.

[2] Then person at position 4 is removed.

[3] Then person at position 1 is removed.

[4] Finally, the person at position 5 is removed.

Hence the person at position 3 survives. (WINNER)

6. **(*) Tower of Hanoi:**

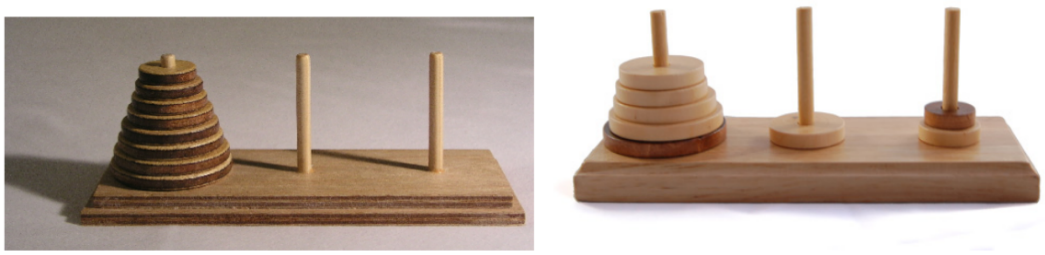
It consists of three rods, and a number of disks of different sizes which can slide onto any rod as shown in Figure 2. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

(a) Only one disk can be moved at a time.

(b) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

(c) No disk may be placed on top of a smaller disk.

With three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2n - 1$, where $n$ is the number of disks.



(a) Starting (b) After Some Steps

Figure 2: Tower of Hanoi Setup

**Task Description:**

You are required to write a $C++$ program which take the number of disks (n) from user and output the set of moves to shift the entire stack to another rod, obeying the above defined rules.

**Input Data and Format**

Enter the number of disks $n$.

**Computation Involved**

(a) Write the recursive solution.

(b) Write the stack based iterative solution.

**Expected Output for Correct and Incorrect Inputs**

For any number of disks output the steps that are required to shift the entire stack in the following format.

(a) Move Disk from Peg A $->$ Peg B

(b) Move Disk from Peg $B->$ Peg C

(c) Move Disk from Peg $A->$ Peg C

(d) . . . .

(e) . . . .

**Sample Output**

$ hanoi

Number of disks : 3

Disk movement sequence to solve it is as follows:

[1] Move disk 1 from peg A to peg C

[2] Move disk 2 from peg A to peg B

[3] Move disk 1 from peg C to peg B

[4] Move disk 3 from peg A to peg C

[5] Move disk 1 from peg B to peg A

[6] Move disk 2 from peg B to peg C

[7] Move disk 1 from peg A to peg C

Total number of steps required to solve this problem are 7.