BLG335E, Analysis of Algorithms I, Fall 2016, Project 3
Handed Out: 04.6.2016
Due: 27.11.2016

Part A. Sorting in Linear Time (50 pts)

1-)(30 pts) In "numbers_to_sort.txt", 1 million 10-digit numbers and their IDs are given. Read N numbers from the file and sort these numbers using Radix Sort. In intermediate sorting step(sorting the numbers on a digit), use Counting Sort. Report the average running times with respect to different N values. **Your program should print the running time and and save the sorted numbers with their IDs in a text file named "sorted_numbers.txt".** What was the complexity you expected to obtain? Show whether this complexity is revealed on the results.

2-)(20 pts) Analyze Gnome Sort and a manipulated version of Freezing Sort[1] given below. Which of them can be used in the intermediate step of Radix Sort? Show all your work. You can benefit from examples and counter examples.

Pseudocode for Gnome Sort[2]

```
function gnomeSort(a[0..size-1])
  i := 1
  j := 2
  while i < size do
    if a[i-1] <= a[i] then
      i := j
      j := j + 1
    else
      swap a[i-1] and a[i]
      i := i - 1
      if i = 0 then
        i := j
        j := j + 1
      endif
    endif
  done
```

C++ Code for Freezing Sort

```cpp
//n is the last indice of the array.
for (int freeze_count = 0; freeze_count < n; freeze_count++)
{
        //Freeze from left
        for (int i = freeze_count; i < (n + freeze_count) / 2 +1; i++)
        {
                if (i != n - i + freeze_count && arr[i] < arr[n - i + freeze_count])
                {
                        int temp = arr[n - i + freeze_count];
                        arr[n - i + freeze_count] = arr[i];
                        arr[i] = temp;
                }
        }

        //Freeze from right
        if (freeze_count != 0)
        {
                for (int i = n - freeze_count; i > (n - freeze_count) / 2; i--)
                {
                        if (i != n - i - freeze_count && arr[n - i - freeze_count] < arr[i])
                        {
                                int temp = arr[n - i - freeze_count];
                                arr[n - i - freeze_count] = arr[i];
                                arr[i] = temp;
                        }
                }
        }
}
```

Your program for this part should run from the command line with the following format:

./studentID_AoA1_P3_1 N

Part B. Heap Sort (50 pts)

In a fictitious online game, there are two clans figthing with each other. Each of the clans consists of 1 million players and they are hierarchically formed as a binary heap. There are some basic rules of the game:

- All players have an ID and a CP (charisma point). CP is used as both an indication of player level and player score.
- The player with the maximum CP in a clan is called the leader of the clan (Position = 0). Other seven players having the maximum CPs are called the henchmen of the leader (Position = 1-7). The remaining players are called soldiers. These titles change automatically according to victories and failures.
- If a clan leader successfully attacks an enemy, she/he captures half of the enemy's CPs. If a henchman attacks an enemy she/he obtains 500 CP and the attacked player loses 500 CPs. If a soldier attacks an enemy, she/he obtains a CP of (height difference between two players+1) multiplied by 30, and the attacked player loses 120 CPs. Always integer division is used.
- All players' CPs must be greater than or equal to 0. If a player takes a damage greater than her/his CP, her/his CP slips down to 0.
- The clan having a greater sum of CPs wins the tournament.

1-)(40 pts) "ClanA.txt" and "ClanB.txt" contains ID and CP information of all players in the clans. Create a class to represent players. For different N values, Read first N elements from "ClanA.txt" to an array of that class and sort them using Heap Sort. Report the average running times with respect to N. Show that complexity of Heap Sort is revealed on the results. **Your program should print the running time and save the sorted player CPs and IDs in a text file named "A_sorted.txt".**

2-)(10 pts) To ease the the problem, suppose that each of the clans contain only 10000 players (Players having IDs between 0-9999). "gamelogs.txt" contains the logs about successful attacks of players in a tournament. Every line in gamelog.txt is defined as follows:

GameID ClanOfAttacker AttackerPosition AttackedPosition

Find out who won the tournament. What are the total CPs of each clan? Save the sorted player CPs and IDs after tournament from Clan A to **"A_results.txt"**.

Your program should run from the command line with one of the following formats:

./studentID_AoA1_P3_2 1 N –For the first question.

./studentID_AoA1_P3_2 2 – For the second question.

All your code must be written in C++ using object oriented approach and it should compile and run on Linux using g++. Do not use STL.

If you have any questions, you can contact Res. Asst. Yusuf Hüseyin Şahin (sahinyu@itu.edu.tr).

[1] http://research.ijais.org/volume2/number4/ijais12-450330.pdf

[2] https://rosettacode.org/wiki/Sorting_algorithms/Gnome_sort