Lab Eight

ID1303: Introduction to Programming

1. Complete exercises from previous sessions.

   Submit the following two exercises as the second programming assignment.

2. Sorting: The sorting problem is to take a sequence of real numbers and to arrange them in non-decreasing order. For example, if the input sequence is 12, 30, -20, 5, 18, 42, -7, 12, then the desired output sequence is: -20, -7, 5, 12, 12, 18, 30, 42.

   Write a program to sort a given sequence $a[1], a[2], \ldots, a[n]$ of integers, using one of the two methods listed at the end.

3. (a) Write a function called printIntersection that accepts two arrays of integers and prints the numbers which are present in both the arrays.

   (b) The file "Network.txt" contains the social-network info of 16 persons: each line contains the name of one person, followed by a unique ID, followed by a list of IDs of that person's friends.

   Write a program to create a structure called **network** which can store a person's name, ID and a list of the person's friends' IDs. Read the file "Network.txt" and store the information in an array of **struct network** variables.

   Accept two persons' IDs from the user and print the names of their common friends (use/modify your printIntersection function from (a)).

   The remaining two exercises are optional.

4. The 15-puzzle is played as follows: There is a 4 by 4 matrix consisting of entries 1,2,...,9,A,B,C,D,E,F in 15 different positions and one empty position. The goal of the game is to start with an initial configuration of the grid and reach the configuration where the characters are in the right order by a sequence of moves, where each move consists of swapping the empty location with any of its neighboring values.

   The matrix on the left is the desirable configuration. This can be reached from the configuration on the right by swapping the blank location value with B and then swapping the blank location value with C.

1	2	3	4
5	6	7	8
9	A	B	C
D	E	F	

1	2	3	4
5	6	7	8
9	A		B
D	E	F	C

Write a program that lets the user play the 15-puzzle. Display an initial configuration and accept a sequence of moves from the user and after each move, display the new matrix. The moves may be accepted as N,W,E or S, where N implies that the blank square has to be swapped with the square above it. W,E,S are self-explanatory. The program will stop if the user inputs the character X. Use zero (0) for the blank location.

5. Two players A and B play a series of N games against each other, each game being independent of the other games. Every game has 3 possible results: Player A wins, Player B wins, or the game is a draw. Each player gains 2 points for a win, 1 point for a draw and 0 points for a loss.

   It is known that in each game, the probability that A wins the game is $p \in [0, 1]$, the probability that B wins is $q \in [0, 1-p]$ and the probability of a draw is $1-p-q$.

   Write a program to accept values for p, q, N, and a number $k \in \{0, 1, 2, \ldots, N\}$ and compute the probability that A wins exactly k games. [Hint: Write a recurrence relation.]

Two simple sorting methods

Method 1: Selection sort

The idea in this method is: find and place the smallest element in the first position, then find the second smallest element and place it in the second position and so on.

How it works: when the first k smallest elements are already placed in positions $a[1], \ldots, a[k]$, the **position** of the smallest element of $a[k+1], a[k+2], \ldots, a[n]$ is found, say this is r.

Then $a[r]$ is swapped with $a[k+1]$ so that now the first $k+1$ elements are the $k+1$ smallest elements in the list.

Placing the correct element in $a[i]$ is thought of as one "pass". Thus there are $n-1$ passes.

On input 12, 30, -20, 5, 18, 42, -7, 12, the method works as below.

Pass 1: The position of the smallest element is 3 (the element is -20), so a[1] and a[3] are swapped.

The list after pass 1: -20, 30, 12, 5, 18, 42, -7, 12

After pass 2: -20, -7, 12, 5, 18, 42, 30, 12

After pass 3: -20, -7, 5, 12, 18, 42, 30, 12

After pass 4: -20, -7, 5, 12, 18, 42, 30, 12

After pass 5: -20, -7, 5, 12, 12, 42, 30, 18

After pass 6: -20, -7, 5, 12, 12, 18, 30, 42

After pass 7: -20, -7, 5, 12, 12, 18, 30, 42

Method 2: Bubble sort

In this method, in one pass, each pair $a[i]$ and $a[i+1]$ are compared and if $a[i] > a[i+1]$, then they are swapped.

At the end of the first pass, the largest element "bubbles" to the end of the list, at the end of the second pass, the second largest element moves to position $a[n-1]$ etc.

On input 12, 30, -20, 5, 18, 42, -7, 12, the first pass is as below:

The list during the first pass:

12, 30, -20, 5, 18, 42, -7, 12

12, 30, -20, 5, 18, 42, -7, 12 (No change in first two positions)

12, -20, 30, 5, 18, 42, -7, 12 (Second and third elements swapped)

12, -20, 5, 30, 18, 42, -7, 12 (Third and fourth elements swapped)

12, -20, 5, 18, 30, 42, -7, 12

12, -20, 5, 18, 30, 42, -7, 12

12, -20, 5, 18, 30, -7, 42, 12

12, -20, 5, 18, 30, -7, 12, 42

The last list is obtained at the end of the first pass, when the element 42 has moved to the end of the list.

The list at the end of pass 2:

-20, 5, 12, 18, -7, 12, 30, 42

At the end of pass 3:

-20, 5, 12, -7, 12, 18, 30, 42

At the end of pass 4:

-20, 5, -7, 12, 12, 18, 30, 42

At the end of pass 5:

-20, -7, 5, 12, 12, 18, 30, 42

With no changes in passes 6, 7. The number of passes guaranteed to work is $n-1$.