Computer Programming 1

Laboratory

- Past exams -

Example: Pattern

- Write a program that checks whether an array of integer ("pattern“) is contained in another array of integers ("text“).
- Randomly generate the two arrays (the use of the rand() function in the library is allowed <cstdlib>) with a size fixed a priori (maximum 100). You can decide the size of the arrays.

```
pattern   →   text

[9, 1, 7] → [4, 5, 9, 1, 7, 3]   ✓

[9, 1, 7] → [5, 1, 2, 6, 3, 3]   ✗
```

1_pattern.cc

Example: Lower to Upper

- Write a program that, taking the names of two files from the command line, copies the first file into the second correcting its syntax.
- In order for a text to be considered correct, the first word of the text, and all words after the characters ".", "?" and "!" must begin with a capital letter.
- You can use the library <fstream>
- You can also write "input >> noskipws;" to prevent the ">>" operator from skipping white space and new lines https://cplusplus.com/reference/ios/noskipws/

text for testing.	→	Text for testing.
please,		Please,
correct me.		correct me.

2-lower_toi_upper.cc

ADT example: @ Post office

- By starting from the implementation of a Queue via linkedlists
  - ./20/queue_c_list.cc
  - ./20/queue_c_list.h
- Write a program that simulates the arrival and disposal of a queue at the post office.
- The elements of the queue are the names of the persons (max 30 characters).
- The main will be a menu with three options:
  1. add person,
  2. dispose of person,
  3. print current situation
- The function strcmp of the library <cstring> can be used

3-at-postOffice.cc

Example: Interleave

- Write a program that interleaves the elements of two initial linked lists alternatively in two different modalities (as explained below).
- The program to be implemented has to:
  1. generate two original linked lists, by filling them with random integer values
     - (*Minimum*) values can be pre-fixed, i.e., not generated at runtime.
  2. create two new linked lists with interleaved elements takes form the two generated lists, as shown in the examples below;
  3. print the values of the initial and interleaved lists in the standard output
- About the notion of "interleaving", giving the following two lists:
  - List1: 7 8 4
  - List2: 13 3 2
  - Examples:
    - An example of interleaved list derived from List1 and List2 is: 7 13 8 3 4 2
    - Another possible example of interleaved list is: 13 8 3 4 2
- Node structure

```
struct Node {
   int data;
   struct Node* next;
};
```

(Example 1)
- Original Lists:
  - List1: 1 3 5 7
  - List2: 2 4 6 8
- The expected output after interleaving the two linked lists alternatively:
  - Interleaved ListA: 1 2 3 4 5 6 7 8
  - Interleaved ListB: 2 3 4 5 6 7 8

(Example 2)
- Original Lists:
  - List1: 14 13 0 1
  - List2: 2 4 6 8
- The expected output after interleaving the two linked lists alternatively:
  - Interleaved ListA: 14 2 13 4 0 6 1 8
  - Interleaved ListB: : 2 13 4 0 6 1 8

Interleave.cpp

Example: Work lists

A workplace attendance management system assigns to each employee a badge with a unique code/id (an integer). Every employee stamps her badge upon entering the workplace, and so on at each exit. The management system keeps track of the entering and exiting stamps of each employee in two different lists (i.e., enter and exist lists). Every time an employee enters/exits, the system adds the identification code/id of the employee at the top of the appropriate list. The same code/id can therefore appear several times within the respective enter/exit list. Ideally, the number of entries and the number of exists should always be the same, or meet a certain condition specified by the giver of work. But every day there are problems, thus the number of entries in the lists can be different.

Overall, three lists are used and maintained by the management system.

- The list of codes/ids of all employees (codes/ids are not repeated);
- The list of integers that contains the codes/ids of the employees who today have stamped their badge when entering in the workspace (each stamp corresponds to a code/id; multiple stamps made by the same employee correspond to the same code/id that appears several times in the list);
- The list of integers contains the codes/ids of the employees who have today stamped their badge when exiting from the workspace (each stamp corresponds to a code/id; more exit stamps performed by the same employee correspond to the same code that appears several times in the list);
- Example
  - List of employees: 12 76 47 23 36 67 16 52 15 14 84
  - List of enter stamps: 84 67 23 76 52 16 67 23 47 12 15 16 67 23 15 16 47 76 12 67 36 23 12 15 52 16 67 36 76
  - List of exit stamps: 84 52 23 14 47 12 84 15 16 47 76 12 52 16 67 36 47 76 12 84 52 36 23 47 76 12 15 16 67 23 76

Assignment

Write a program that takes the three linked lists of integers as arguments: employees, enter stamps, and exit stamps, and produces in output a linked list of representative integers that correspond to (unique) codes/ids of employees such that their number of enter stamps is different than their number of exit stamps

Example (for the lists as examples),
Warning list: 67, 15, 23, 16, 84, 47, 52, 76, 12, 14

Office_v3.cpp

Example: Submatrix

- Write a program that, given two integers in input (rows and columns), dynamically creates an array and fills it with random integer values (the use of the rand() function in the library is allowed <cstdlib>).
- Next, ask the user about the size of the submatrix to print.
- The dimensions are provided by four numbers, corresponding to the coordinates of the first element (2 numbers) and the size of the submatrix (number of rows and columns).
- If the size provided by the user exceeds that of the array, print the maximum available submatrix.
- Finally, deallocate the matrix.

```
1 2 9 1
5 6 7 8
9 5 7 2
7 2 9 3
```

```
input: 2,1,2,3  =>
```

```
5 7 2
2 9 3
```

2-submatrix.cc

Example: File (part 1)

Write a C++ program that implements the following functions.

(1) Function one.
Write a first function that creates Y arrays with elements of type integer, and fill them with valid numbers that correspond to the length of strings read from a file. Implement one of the following two alternatives

- (***minimum***) Y and the numbers in each array are fixed
- (***best***) Y the numbers are read from a file named input.txt (example in the box) composed as follow:
  - The number of rows of the file represents Y.
  - Each row contains X strings (assumption: no white space at the begin and end of each row in the file).
  - Example

In such an example of input.txt, we have:

- Y: 4
- X: 5 and numbers: 3, 5, 3, 8, 4, 7
- X: 8 and numbers: 3, 6, 3, 8, 4, 4, 4, 9, 8
- X: 11 and numbers: 2, 1, 5, 4, 4, 8, 4, 3, 3, 9,
- X : 15 and numbers: 7, 5, 3, 2, 7, 4, 13, 2, 1, 8, 3, 5, 4, 8, 10

Example of input.txt content

The first row contains five strings
The second row contains more than five different elements
At a given time Jean realizes that she has forgotten something
Reading helps you to improve your understanding of a language and build your personal vocabulary

(2) Function two
Write a second function that finds and prints all distinct elements of a given array of integers.

- Example,
  - Input array: 3, 5, 3, 8, 4, 7
  - Unique elements of the said array: 3, 5, 8, 4, 7

Example: File (part 2)

(3) Function three
Write a third function that, given an array of integers, finds the number of pairs of integers in the array whose sum is equal to a specified number, if any. Note, the order of the numbers in the pair is not relevant.

- Examples in the box -->

Input array: 3, 5, 3, 8, 4, 7
- Array pairs whose sum equal to: 7 (last element of the array):
- Number of pairs whose sum equal to 7: 1
  - 3, 4

Input array: 3, 6, 3, 8, 4, 4, 4, 9, 8
- Array pairs whose sum equal to: 8 (last element of the array):
- Number of pairs whose sum equal to 8: 1
  - 4, 4

Input array: 2, 1, 5, 4, 4, 8, 4, 3, 3, 9, 9
- Array pairs whose sum equal to: 9 (last element of the array):
- Number of pairs whose sum equal to 9: 2
  - 1, 8
  - 5, 4

Input array: 7, 5, 3, 2, 7, 4, 13, 2, 1, 8, 3, 5, 4, 8, 10
- Array pairs whose sum equal to: 10 (last element of the array):
- Number of pairs whose sum equal to 10: 3
  - 7, 3
  - 5, 5
  - 2, 8

(4) Function four
Write a fourth function that prints the ordered numbers in a given array of integer values. Implement one of the following two alternatives:

- (***accepted***) use an algorithm to order
- (***best***) use the Bubble sort to order

(5) Function five
Write a program that applies the functions (2), (3) and (4) iteratively to the Y arrays created in function (1) and prints the results in the standard output and in a file (named output.txt)

Implementation constraints:
- Do not use complex data structures such as Queue, Stack, Vectors.
- (optional) Use at least two functions with different types of parameter passing.
- (optional) Do not use global variables.

file_array.cc

Example: Trees (part 1)

(1) Write a C++ program to implement a Binary Tree, traverse it in pre/post-order, and compute the diameter of the tree.

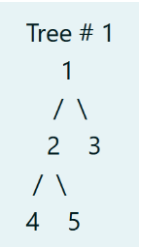
Implementation request:

(1) Develop a first function that builds a Binary Tree (BT)
Use the following structure for the node for the tree

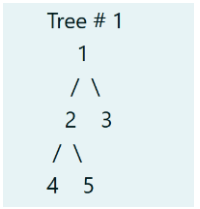
```
struct Node {
  int data;
  Node * left;
  Node * right;
}
```

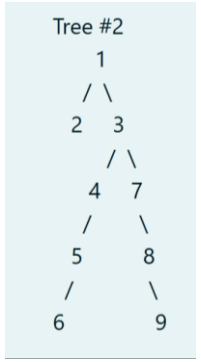
(a) Develop only one of these alternatives:

(min) the function builds the following tree, and turn back the link to the root element



(best) the function builds the following two trees, and turn back two links to their root element





Example: Trees (part 2)

(2) Develop a second function that crosses the built BT and prints the value of the field data in pre-order

(3) Develop a third function that, given the root of the BT, returns the high of tree

(4) Develop a fourth function that, given the root of the BT, returns the length of tree diameter and the value of the field data of source and target nodes of such diameter.

- The Diameter of a BT is the longest distance between any two nodes of that tree. Note that this path may or may not pass through the root.
- Examples
  - Tree # 1: diameter = 4, first node = 4, second node = 3
  - Tree # 2: diameter = 7, first node = 6, second node = 9

Implementation constraints:

• Do not use complex data structures such as Queue, Stack, Vectors.
• (optional) Use at least two functions with different types of parameter passing.
• (optional) Do not use global variables.

trees.cpp