Algorithms and Data Structures
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-231-A
February 3rd, 2020

Homework 1

- Submit one ZIP file per homework sheet which contains one PDF file (including pictures, computations, formulas, explanations, etc.) and your source code file(s) with one makefile and without adding executable, object or temporary files.
- The implementations of algorithms has to be done using C, C++, Python or Java.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/ch_231_a/2020_1/Grading_Criteria_ADS.pdf

Problem 1.1 *Template array search* (1 point)

Language: C++

First write a generic `... array_search(...)` function using templates. The array search function should receive an array of elements, the corresponding number of elements, the element searched for and should determine if that element is in the array or not by returning its index or -1 if not in the array. Then write a test program with a `main` function which uses the function above for multiple types (some example basic data types and `Complex` objects).

You can assume that the array will contain at least one element. Make sure that your code works not only with basic data types but also with a simple `Complex` class.

Problem 1.2 *Stack with templates* (2 points)

Language: C++

Design and implement a generic stack class. By using templates your stack has to be able to store any type of data. The underlying data structure should be an array.

While designing and implementing your class you cannot use any of the container classes from the Standard Template Library (STL).

Your class should support the following operations (read all the requirements before starting to write code):

- `Stack()`: this constructor initializes the stack to a size of 10.
- `Stack(const Stack&)`: copy constructor, create a real copy of the stack.
- `Stack(int size)`: this constructor sets the stack's size to the given size.
- `bool push(T element)`: adds `element` to the top of the stack. It should return `true`, if the element has been successfully pushed. So as long as you do not provide an `extend()` method, you need to check for available space and return `false` if there is no more space.
- `bool pop(T& out)`: pops an element from the top of the stack. The element is put into `out`. It should not crash if there are no elements on the stack, but rather return `false`, otherwise it should return `true`.
- `T back(void)`: returns the data on the top of the stack, without changing the stack.
- `int getNumEntries()`: returns the number of entries of the stack at a given moment.
- Destructor for the template class.

Finally, write a simple program which tests the functionality of your stack.

Name the files `Stack.h` and `testStack.cpp`.

You can assume that the input will be valid.

Problem 1.3 *List with templates as doubly linked list* (2 points)

Language: C++

Implement a generic list using templates using a doubly linked list instead of an array. Provide constructors, destructor and methods for returning (with and without remove) the first and last element as well as adding a new element at the front and at the end of the list. Also implement a method which returns the amount of elements in the list.
Name the files `LinkedList.h` and `testLinkedList.cpp`.
You can assume that the input will be valid.

Problem 1.4 *Using vectors I* (1 point)

Language: C++

Write a program which does the following using a `vector` object:

1. Read words from the keyboard (one per line) until the entered word is equal to the word "`END`". You can assume that the words do not contain whitespace.
2. Insert these words into the vector at the end (excluding the word "`END`").
3. Print the words on the standard output separated by spaces using the index operator.
4. Print the words on the standard output separated by comma using a corresponding iterator. Make sure that you do not print a comma after the last word.

You can assume that the input will be valid.

Problem 1.5 *Using vectors II* (1 point)

Language: C++

Write a program which does the following using a `vector` object:

1. Read strings from the keyboard (one per line) until the string is equal to the string "`END`". Make sure that you can read strings which contain spaces and/or tabs as well.
2. Insert these strings into the vector at the end (excluding the string "`END`").
3. If the second and fifth elements (i.e., strings) exist, swap the second and fifth element. If one of them or both do not exist then print a message on the standard output.
4. Replace the last element with the string "`???`".
5. Print the strings on the standard output separated by comma using the index operator. Make sure that you do not print a comma after the last string.
6. Print the strings on the standard output separated by semicolons using a corresponding iterator. Make sure that you do not print a semicolon after the last string.
7. Print the strings on the standard output in the reversed order separated by space using an iterator.

You can assume that the input will be valid.

How to submit your solutions

You can submit your solutions via *Grader* at **`https://grader.eecs.jacobs-university.de`** as one generated ZIP file containing one PDF file and source code files and makefile.
If there are problems with *Grader* (but only then), you can submit the file by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that starts with CH08-320201**.

Please note, that after the deadline it will not be possible to submit solutions. It is useless to send solutions by mail, because they will not be graded.

This homework is due by Monday, February 10th, 23:00.