technische universität dortmund

Department of Computer Science
Chair 11 / Algorithm Engineering

Dr. Carsten Gutwenger

Winter 2011/12

Object-oriented Programming
Assignment Sheet No. 10

Date: January 10

Exercise 10.1 (Pointers)

Read and understand the program below, which works with pointers.

```
#include <iostream>

using namespace std;

int main()
{
  int a,b,c;
  int *ptr1, *ptr2, *ptr3;

  ptr1 = &a; ptr2 = &b; ptr3 = &c;
  *ptr1 = 1; *ptr2 = 2; *ptr3 = 3;
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl; // *CHECK1*

  ptr1 = &c; ptr3 = &a;
  *ptr1 = 4; *ptr3 = *ptr3 - 1;
  cout << "a = " << a << ",  b = " << b << ", c = " << c << endl; // *CHECK2*

  *ptr3 = *ptr1 + *ptr2; ptr1 = ptr2;
  *ptr1 = *ptr1 + 4; *ptr2 = *ptr2 + 3; *&c = *&a * 2;
  cout << "a = " << a << ", b = " << b << ", c = " << c << endl; // *CHECK3*

  return 0;
}
```

Fill the table with the values of the variables a, b, and c at the three checkpoints. Verify your answers by running the program.

	a	b	c
CHECK1			
CHECK2			
CHECK3			

Exercise 10.2 (Dynamic Memory Allocation: Stacks)

A *stack* is a last in, first out (LIFO) data structure. In this exercise, we want to implement a stack storing `double` numbers. Our `Stack` shall provide the following methods:

- `void push(double x)`

  adds a new number `x` to the top of the stack.

- `double pop()`

  removes the top-most number from the stack and returns it.

- `double top() const`

  returns the number on the top of the stack.

- `int size() const`

  returns the number of elements stored in the stack.

The `pop` and `top` methods cannot be applied to an empty stack; in such a case an error message should be displayed. Use the following structure for representing elements on the stack:

```
struct StackElement {
    double m_value;        // number stored on stack
    StackElement *m_next;  // points to element on stack below this one
                           // (0 if this is bottommost element)
    StackElement(double x, StackElement *pNext)
        : m_value(x), m_next(pNext) { }
};
```

Write a class `Stack` that implements a stack of `doubles`. Do not use any container classes from the C++ standard library!

Exercise 10.3 (The RPN Calculator)

Reverse Polish notation (RPN) is a mathematical notation, where operators follow their operands. E.g. the expression `(1+2)*(3+4)` in RPN is `1 2 + 3 4 + *`. Expressions in RPN can easily be evaluated using a stack of numbers: Whenever we encounter a number, we push it on the stack, when we encounter an operation, we pop two numbers from the stack, apply the operation and push the result on the stack.

Write a program that implements an interactive RPN calculator. Use the stack implemented in Exercise 10.2. Your program should provide (at least) the following functionality:

- Enter a number, which is then pushed on the stack.
- Enter an operation (`+`, `-`, `*`, `/`), which then pops two numbers from the stack, applies the operation, and pushes the result on the stack.
- Display the top-most number on the stack.
- Extend your program such that functions are also supported (e.g. `sqr`, `exp`, `ln`); the function shall then be applied to the top-most number on the stack.