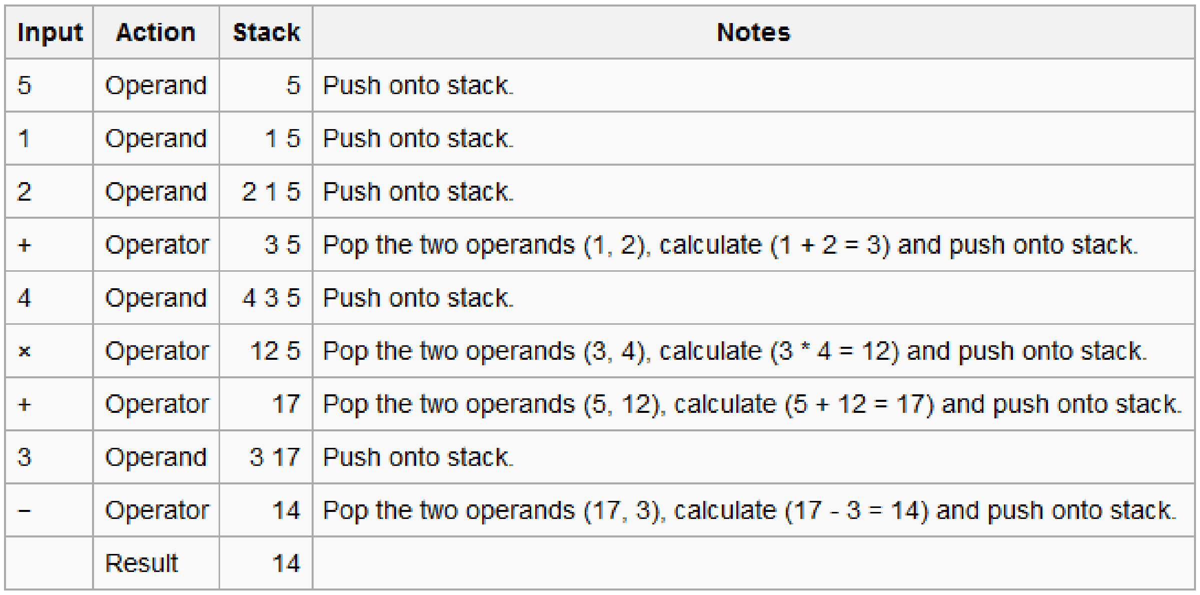
Metropolia
C++ Programming
teacher: Pasi Ranne

12/08/2016
Exam

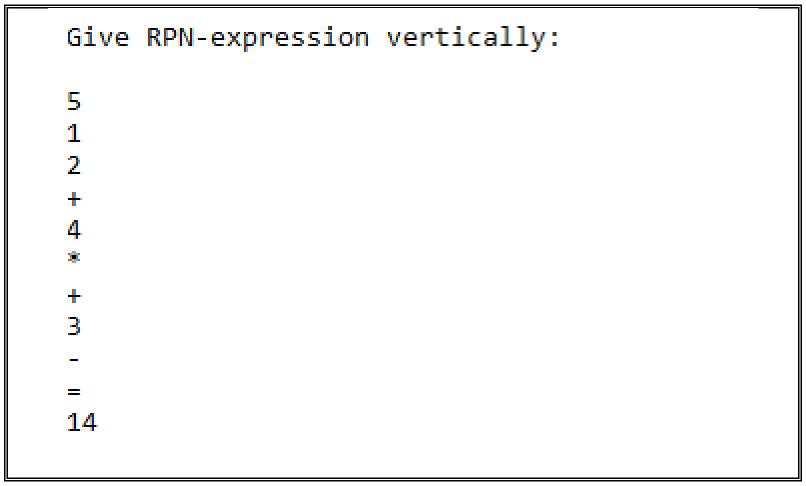
Return all your files 1.cpp, 2.cpp, 3.cpp and 4.cpp in Tuubi before deadline!

1. The principle of **RPN** calculator (Reverse Polish Notation calculator) has been described in address https://en.wikipedia.org/wiki/Reverse_Polish_notation (figure 1). Implement program **1.cpp** which works as **RPN** calculator. You have to use stack and you can assume that all numbers are one digit integers (0, 1, 2, 3, 4, 5, 6, 7, 8 and 9). In figure 1 is example how it works in theory. When you enter integer you have to push it in stack. When you enter operand (+, - or *) then you have to pop two numbers from stack and make the arithmetic operation. The result you have to push in stack. When you enter character '=' you have to end and print the result of expression. Test your program with RPN expression **5 1 2 + 4 × + 3 –** which infix expression is "5 + ((1 + 2) × 4) − 3". Sample input and output is in figure 2. (**6 points**)



Input	Action	Stack	Notes
5	Operand	5	Push onto stack.
1	Operand	1 5	Push onto stack.
2	Operand	2 1 5	Push onto stack.
+	Operator	3 5	Pop the two operands (1, 2), calculate (1 + 2 = 3) and push onto stack.
4	Operand	4 3 5	Push onto stack.
×	Operator	12 5	Pop the two operands (3, 4), calculate (3 * 4 = 12) and push onto stack.
+	Operator	17	Pop the two operands (5, 12), calculate (5 + 12 = 17) and push onto stack.
3	Operand	3 17	Push onto stack.
−	Operator	14	Pop the two operands (17, 3), calculate (17 - 3 = 14) and push onto stack.
	Result	14	

Figure 1. Principle and example of RPN calculator

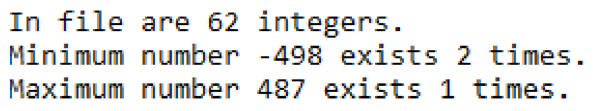


```
Give RPN-expression vertically:

5
1
2
+
4
*
+
3
-
=
14
```

Figure 2. Sample input and output

2. Download http://users.metropolia.fi/~pasitr/2016-2017/TI00AA50-3010/exam/A.txt and implement a program **2.cpp** which counts how many times the largest and the smallest number occurs in the file **A.txt**. Sample output is in figure 3. (**6 points**)



```
In file are 62 integers.
Minimum number -498 exists 2 times.
Maximum number 487 exists 1 times.
```

Figure 3. Sample print of program 1.cpp

3. **MVC pattern**. In link https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm you can see the principle of MVC pattern which is programmed in Java. In the address http://users.metropolia.fi/~pasitr/2016-2017/TI00AA50-3010/kt/11/solutions/1.cpp is one solution with C++. Now you have to implement throwing dice program **3.cpp** where you use MVC pattern. In a dice is six symmetric size faces (figure 4). Probability of each faces is same (1/6). Use randomize function of C++. First you can enter how many times you throw dice. Sample print of program is in figure 5. (**6 points**).

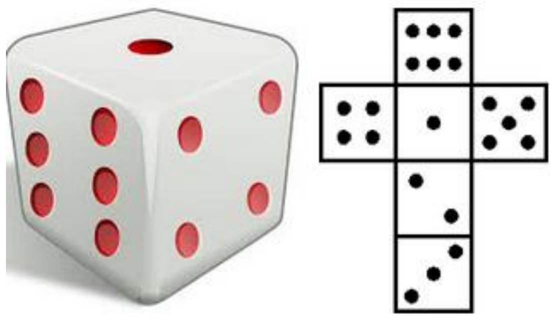
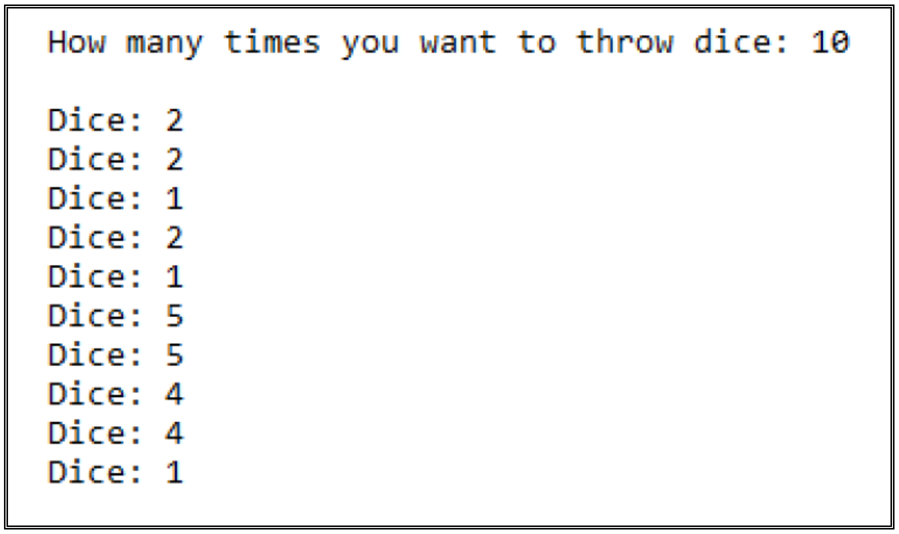


Figure 4. Dice contains 6 faces

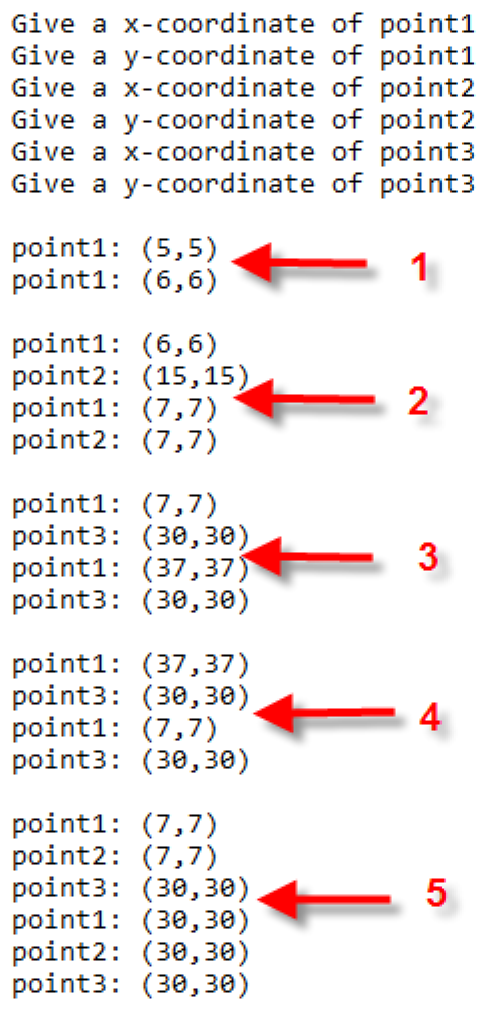


```
How many times you want to throw dice: 10

Dice: 2
Dice: 2
Dice: 1
Dice: 2
Dice: 1
Dice: 5
Dice: 5
Dice: 4
Dice: 4
Dice: 1
```

Figure 5. Sample print of program 3.cpp

4. **Operator overloading**. Implement program **4.cpp** where you define class **Point**. Class **Point** contains two attributes **x** and **y**. This program contains four additional overloaded operators to the class **Point**. Those operators are +, –, = and ++. After you have implement these operators you have to implement main function. First in main program retrieves coordinates. First operation is **++point1** (figure 6, point 1). Second operation is **point2 = ++point1** (figure 6, point 2). Third operation is **point1 = point1+point3** (figure 6, point 3). Fourth operation is **point1 = point1-point3** (figure 6, point 4). Fifth operation is **point1 = point2 = point3** (figure 6, point 5). Sample input and output is in figure 6. (**6 points**)



```
Give a x-coordinate of point1: 5
Give a y-coordinate of point1: 5
Give a x-coordinate of point2: 15
Give a y-coordinate of point2: 15
Give a x-coordinate of point3: 30
Give a y-coordinate of point3: 30

point1: (5,5)
point1: (6,6)

point1: (6,6)
point2: (15,15)
point1: (7,7)
point2: (7,7)

point1: (7,7)
point3: (30,30)
point1: (37,37)
point3: (30,30)

point1: (37,37)
point3: (30,30)
point1: (7,7)
point3: (30,30)

point1: (7,7)
point2: (7,7)
point3: (30,30)
point1: (30,30)
point2: (30,30)
point3: (30,30)
```

Figure 6. Sample input and output

Good luck for the exam!