TND012/2017: Dugga instructions

Attention

This is an **individual** home dugga.

Books, notes, and internet material can be used.

If you find any open questions in some of the presented exercises then feel free to make your own decisions. However, your decisions should be reasonable and must not contradict any of the conditions explicitly written for the exercise. Please, write comments in the programs that clarify your assumptions/decisions, if any.

If you use Code::Blocks then follow the instructions given in the appendix of Lab 1. Remember that you should not ignore the compiler warnings, since they are usually an indication of serious problems in the code.

Course goals

This is an assessment of whether you can

- write simple programs, with basic input and output,
- use selection statements, and
- use simple loops.

All the points above have been discussed in lectures 1 to 4 and, lessons 1 and 2.

Requirements for grade G

- Readable and well indented code.
- Use of good programming practices.
- Global variables cannot be used.
- Respect for submission instructions.
- Code copied from a web page should be clearly indicated through a commented line containing the web page address used as source, with exception for the material posted on the course website.
- Use of statements that are not part of the ISO C++ leads to automatically failing the dugga.
- Your programs must pass all test examples shown in this paper.
- Other criteria are listed in the feedback report document available from course website.

Note that there is no guarantee that your code is correct just because it passes all given test examples. Thus, you should always test your code with extra examples, you come up with.

Deadline

11 of September, 12:00.

Submission instructions

1. Create **one** source (`.cpp`) file for the exercise. The file must be named with your LiU login (e.g. `ninek007.cpp`).
2. Write your name and personal number at the beginning of the source code file.
3. Submit only the source code file (`.cpp`) through Lisam, without compressing it (i.e. neither `.zip` nor `.rar` files are accepted).

Remember that you should deliver only the source code file. Moreover, answers to the dugga exercises sent by email are ignored.

Duggor solutions submitted not accordingly the submission instructions are ignored.

Questions

The aim of the dugga is that you solve the exercise without help of other people. However, we give you the possibility to send us questions about the problem in this dugga by email. If you are a MT student then you should email Aida Nordman (aida.vitoria@liu.se). If you are an ED or KTS student then you should email Martin Falk (martin.falk@liu.se).

Only emails received from 12:00 to 20:00 on Friday will be answered. Emails received after 20:00 on Friday are not answered. The emails will be answered until Saturday at 12:00.

Be brief and concrete. Emails can be written either in Swedish or English.

Note that you should not send emails related to Lisam system.

User support for Lisam

Help and support for Lisam system is available at helpdesk@student.liu.se and by calling the phone number 013-28 58 98.

Login and password to access the course material

All course material, like lecture slides and code examples, are available through the course website (**`http://weber.itn.liu.se/~aidvi05/courses/10/index.html`**). However, the material is password protected. Use the following login and password to access the material.

login: `TND012`   password: `TND012ht2_12`

Lycka till!!

Exercise

Write a program that calculates the price to pay for a one-day renting of a car. The price should be calculated according to the following rule.

- The user pays for every 10 kilometers that she/he drives the car. For the first 50 kilometers, each 10 kilometers costs 125 sek. After 50 kilometers, each 10 kilometers costs 200 sek.

Note, for example, that if the user has used the car for 5 kilometers then she pays 125 sek. Likewise, note that if she has driven 56 kilometers then she will pay 825 sek (the last 6 km cost as much as 10 km).

The program starts by asking the number of kilometers marked by the car before the renting and after the renting (two integer values). Then, it displays the price to be paid.

Your program should validate the user input. For instance, the kilometers entered by the user cannot be negative. If the user input is not valid then an error message should be displayed and the user should be able to enter the input again.

Assume that the user always enters integers as input for the program.

Several examples are given below. Values in green color are entered by the user.

Example 1

```
Starting Kms? 20
Ending Kms? 25

Price = 125 sek
```

Example 2

```
Starting Kms? 100
Ending Kms? 156

Price = 825 sek
```

Example 3

```
Starting Kms? -50
Ending Kms? 75

Invalid input!!

Starting Kms? 50
Ending Kms? 75

Price = 375 sek
```

Example 4

```
Starting Kms? 10150
Ending Kms? 10235

Price = 1425 sek
```

Example 5

```
Starting Kms? 10150
Ending Kms? 10177

Price = 375 sek
```

Example 6

```
Starting Kms? 1200
Ending Kms? 1100

Invalid input!!

Starting Kms? 1200
Ending Kms? 1305

Price = 1825 sek
```