TND012/2017: Dugga 3

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.

You are not allowed to post any information about the dugga in any website until the deadline expires.

Course goals

This is an assessment of whether you can structure a program using functions.
Lectures 1 to 8, lessons 1 to 5, and labs 1 to 5 are relevant for this dugga exercise.

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

9 of October, 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.

It is the last submission made in Lisam that is graded. All other submissions you may have uploaded in Lisam are not considered.

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. To this end, email Aida Nordman (aida.vitoria@liu.se). Your emails must have the course code and your study programme in the subject (e.g. "`TND012/KTS1:` ...").

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

Assume that a company associates codes with the goods it delivers. These codes are positive integers (i.e. integers larger than zero). It is possible that these codes are unintentionally modified, e.g. a sensor may read the wrong code assigned to a product. Therefore, the company has introduced a simple policy to test whether a code is valid. This policy is described below.

Consider for instance the code **3**6**6**8**9**7**2**8**0**0**6**6**9**1.

1. Beginning from the right, double every second digit (i.e. digits highlighted in bold). If doubling results in a two digit number then add the single digits of the two digit number to get a single digit result. For the code above we have: [9x2=18, 1+8=**9**], [6x2=12, 1+2=**3**], [0×2=**0**], [2×2=**4**], [9×2=18, 1+8=**9**], [6×2=12, 1+2=**3**], [3×2=**6**]. These single digit results, which are highlighted in red, are then summed: [9+3+0+4+9+3+6=**34**].
2. Add all the digits of the code not considered in step 1. In our example: [1+6+0+8+7+8+6=**36**].
3. Sum the results from step 1 and step 2. In our example, we have: [34+36=**70**].
4. If the result from step 3 is divisible by 10 then the code is considered valid. Thus, the code above is valid, since **70** is divisible by 10.

Write a program that first reads a list of product codes. It should then display a two-column table, where the first column lists all valid codes and the second column lists all invalid codes.

Assume the user only enters positive integers without leading zeros (i.e. the user does not enter integers such as 007). Moreover, the user indicates the end of the list of codes by typing a non-numeric value (e.g. "`STOP`").

Your program **must** satisfy the following coding requirements.

- The program must be structured with the help of functions.
- The `main` function of your program must not contain any loops.
- There must be a separate function, named `read_codes`, that reads the user entered codes from `cin` and stores them in an array of `long long ints`.
- There must be a separate function, named `check_code`, that given a code (`long long int`) tests whether the code is valid according the steps given above.
- There must be a separate function, named `display_table`, that displays the two-column table.

Note that you decide which arguments the functions need and what they should return, without going against the given instructions. Feel free to add any other functions, besides the ones requested above.

An example is given below. Values with green color are entered by the user.

Hint: Separate the input codes in two groups, one for the valid codes and another for the invalid codes. Store each group in its own array of `long long int`s.

Example 1

```
Enter the codes:
5827
445511
33415
32326
59550
1111
59551
5821
33326
445522
4388576018410707
36689728006691
36453624731948
36951397403587
3537722836097167
3542870005189810
30290496911125
30082862197490
30243968373506
STOP

               Valid codes            Invalid codes
          =============================================

                     5827                    445511
                    33415                      1111
                    32326                     59551
                    59550                      5821
         4388576018410707                     33326
           36689728006691                    445522
           36453624731948          3542870005189810
           36951397403587
         3537722836097167
           30290496911125
           30082862197490
           30243968373506
```