

Homework 1

Due: 04.03.2012 at 23:59

“Who Wants to Be a Millionaire?” – Part 1

1. Introduction

The purpose of the homework assignment is to develop the well-known game called “Who Wants to Be a Millionaire?”. For detailed information about the game check Wikipedia (http://en.wikipedia.org/wiki/Who_Wants_to_Be_a_Millionaire%3F).

NOTE! Our version of the game will have several variations from the original scheme described in Wikipedia. Homework specifications will reveal the exact requirements.

The assignment is divided into two parts. Homework 1 includes preparation of data for the game. Homework 2 requires implementation of the specified gameplay. Detailed description of tasks for Homework 2 will be provided in the Homework 2 specification. The tasks for Homework 1 are described below.

2. Description of the Game

In our version of the game the contestant starts the game from the "hot seat" (the term explanation can be found in the above-mentioned link) and answers the asked questions one by one. Questions are multiple choice questions with four answer options (labelled A, B, C and D) and only one option is correct. There is no time limit to answer a question.

Each question has a difficulty level and a corresponding prize. The contestant gets one question from each difficulty level. There are 12 levels. Questions increase in difficulty, so subsequent questions are played for larger prizes. After viewing a question, the contestant can quit with the money s/he has already won or can answer.

The complete sequence of prizes is as follows:

€500
€1,000
€2,000
€4,000
€8,000
€16,000
€32,000
€64,000
€125,000
€250,000
€500,000
€1,000,000

Here the specification needed for this stage ends. More description of the gameplay will be provided in the specification of Homework 2.

3. Your Tasks

Task 1

Find out the main objects in the game. Roughly decide the properties (data members and member functions) of these objects.

Task 2

- Create a question class.
- Create a list of question objects.
- Create a set of question (12 questions) for the contestant by randomly choosing a question of a given difficulty level from the question list. (*Hint:* Think of how to extract the questions of a given difficulty and then randomly choose 1 from them. Example for generation of a random number can be found in the **6. Generating Random Numbers** section.)

Write a program that supports the following functionalities:

- Read the questions from a text file.
- Add/Remove/Edit a question to/from the question list.
- Search the questions on basis of a small set of word given by user and then Print/Edit a question to/from the question list.
- Selects a question from question list and then Remove/Edit a question to/from the question list and save question to a file.
- Print a question on the screen.
- Save the question list to a text file.
- Generate a question set for the player. Print the set on the screen.

Submit .hh, .cc, .txt files generated in the Task 2 according to requirements specified in the course webpage.

4. Implementation Requirements

Use STL for creating a list of questions.

Give the text file as an argument to the main function.

5. Other Requirements

For submission and style requirements check the course homework webpage at <http://www.cs.tut.fi/~prog2/homework/>.

6. Generating Random Numbers

The following piece of code is an example of generating random numbers in the [1, upperBound] range:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    int upperBound = 5;
    srand(time(NULL)); // Initializes the random number generator.

    int randomNumber = rand()%upperBound + 1; // Formula to get the random
                                                // number in the [1, upperBound]
                                                // range
    cout << randomNumber << endl;
}
```

Explanation of the functions used in the code:

Function Name	Explanation	Library
<code>int rand (void);</code>	<u>Generates random number.</u> Returns a random number in the range 0 to RAND_MAX. RAND_MAX is a constant defined in <stdlib>.	<stdlib>
<code>void srand (unsigned int seed);</code>	<u>Initialize random number generator.</u> The random number generator is initialized using the argument passed as seed. In order to generate random-like numbers, srand is usually initialized to some distinctive value, like those related with the execution time. For example, the value returned by the function time (declared in <ctime>) is different each second.	
<code>time_t time (time_t * timer);</code>	<u>Gets current time.</u>	<ctime>

7. Example User Interface

Running the executable

```
./hw1 hw1_123456.txt
```

Sample interface

```
Text file successfully loaded.
Number of questions - 28

Select menu item:

[1] Add a new question
[2] Search a question
[3] Select
[4] Print
[5] Generate a Question Set
[6] Save questions to a file
[7] Print the generated Question set
Your choice:
```

Each of the menu items might print a set of queries that are specific to that menu item. Find out yourself what the queries would be. The following is the example of add a question.

[1] Add a question

```
Enter question description: A bear walks one mile south, turns left and walks
one mile east and then turns left again and walks one mile north and arrives at
its original position. Which adjective describes the bear the best?

Enter answer A: sleepy
Enter answer B: angry
Enter answer C: white

Enter Difficulty level: 9

Question Added successfully...

Press any key to return to start menu...
```

[2] Search a Question:

```
    Please enter a word to search: what

    Questions containing the string "what" are: 1 3 16
    [A>Edit questions
    [B>Print questions
    [C>Return to start menu...
Your choice: A
Enter a Question number to edit: 16
Editing question #16
-----
Here the current content of the question #16 is printed(difficulty level, question, etc.)
-----
Enter question description: A bear walks one mile south, turns left and walks
one mile east and then turns left again and walks one mile north and arrives at
its original position. Which adjective describes the bear the best?

Enter answer A: sleepy
Enter answer B: angry
Enter answer C: white

Enter Difficulty level: 9

After editing it, provide the same options.
Questions containing the string "what" are: 1 3 16
[A>Edit questions
[B>Print questions
[C>Return to start menu...
Your choice:
```

[3] Select a Question:

```
Select question number [1-28]: 54
ERROR: Question number should be from 1 to 28
```

```
Select a question number [1-28]: 16
```

Here the current content of the question #16 is printed (difficulty level, question, etc.)

```
[A] Edit question
[B] Remove question
[C] Save question to file
[D] Return to start menu...
```

Your choice: A

Editing question #16

Enter question description: A bear walks one mile south, turns left and walks one mile east and then turns left again and walks one mile north and arrives at its original position. Which adjective describes the bear the best?

Enter answer A: *sleepy*

Enter answer B: angry

Enter answer C: white

Enter Difficulty level: 9

After editing it provides the same options .

```
[A] Edit questions
[B] Print questions
[C] Return to start menu...
```

Your choice:

[4] Print:

```
[A] Print all questions
[B] Print a question by number
[C] Print questions of a given difficulty level
[D] Return to start menu...
```

Your choice: B

Enter a Question number [1-28]: 16

Question 16

A bear walks one mile south, turns left and walks one mile east and then turns left again and walks one mile north and arrives at its original position. Which adjective describes the bear the best?

Option A: *sleepy*

Option B: angry

Option C: white

After printing it provides the same options .

```
[A] Print all questions
[B] Print a question by number
[C] Print questions of a given difficulty level
[D] Return to start menu...
```

Your choice:

Minimum requirements to Pass Homework Assignment:

- Homework assignments must be implemented with C++, no C.
 - The ADTs identified
 - The properties (operations and attributes) of the ADTs
 - The data representation used

- The programs must compile without warnings and errors with **tutg++**.
- The program works according to the specification. Note! also unnecessary extra features may reduce the grade at least **90%** style grade from the style checker
- **No C-code:** No goto, printf etc. No global variables.
- Programs are commented properly. Each file starts with a comment stating the name and student number of the programmer and briefly describes the contents of the file
- The names of the functions and variables make sense and the program is readable.
- The program is well structured
- Classes are well defined

- Submit your files as a .ZIP/.RAR files or directly attaching files to the mail.

Submit the source code of your program to prog2@cs.tut.fi as an attachment in an email. The name of the code file must be [student number].cc where the [student number] is your own student number, e.g. 123456.cc. The subject of the email should be HW1: [student number], e.g. HW1: 123456.

The deadline

The deadline for this assignment is Sunday the **4th of March 2012 at 23:59:59**. Late submissions are automatically rejected if no arrangements have been made with the lecturer.

Note: Deadlines are set to Sundays at midnight.

Good Luck!