Exercise 4 – Dice Gambling Game

DATE DUE: start of class 11

Focus on Arrays, Strings, Game Loop.

Algorithm Requirement: (pseudocode, plan) Due: Class 10 Hard copy to be handed in, in class.

Hand in through the Dropbox
Make sure your files are working with Visual Studio 2010!

Dice Gambling Game

Write a Dice Gambling program with the following features

1. Keep track of the player's money
2. The game consists of a series of rounds. Each round the following things happen:
    a. The Player is asked if they want to keep playing or quit
    b. If the Player wants to play, they are asked how much they want to bet.
    c. A round of the game is played (exact rules determined by you!)
    d. Player money is adjusted and reported
3. If the player runs out of money or decides to quit then the game is over – report the results and then exit.

Things to remember

1. Conform to the coding standards!
2. Use only language features we've learned so far: Variables, Operators, Constants and Enums, Truth Statements and flow control (if/else statements), Looping (while, do/while, for), random numbers, strings, and arrays or vectors.
3. Program must contain a game loop.
4. You may use existing dice games; you don't have to create an original game.

Submission

1. Compress your Visual Studio 2010 project directory into a zip archive. (Under Visual Studio 2010/Projects/WhateverYouCalledIt, not the entire "Projects" directory but the WhateverYouCalledIt directory).
2. Rename your zip file as follows: *LastnameFirstnameExercise1.zip* (replacing "*LastnameFirstname*" with your names).
3. Copy this file to your dropbox.

DO NOT RENAME YOUR PROJECT DIRECTORY, just your zip file. The IDE uses the project name internally and your .sln file will not work properly if you rename the project even if your .cpp file is fine.

Grading Guidelines

Plan – Pseudocode (10 points)

- 10 points will be for the pseudocode. If it is late (see due date above), 0 points.

Design and Debugging - Functionality (40 points)

- 40 points if the program runs without bugs as described above. Proper use of a Game Loop.
- 30 points if the program compiles, does not crash, displays some of the behavior but has some minor bugs.
- 15 points if the program compiles, but either crashes part way or has some major bugs.
- 0 points if the program does not compile, or compiles but crashes immediately.

Coding Standards/Readability/Comments (20 points)

- 20 points if variables used throughout the program are declared at the beginning, while single-use temporary variables are declared immediately before use, code for similar statements (such as prompting the player for input) are written in a consistent style, variables are named appropriately, whitespace and indentation is used consistently. The ***code is easy to read***.
- 10 points if there are a few minor style errors. Things like variable names not making clear what they contain, lines written or indented inconsistently, different parts of the program not separated by line breaks. ***Code is fairly easy to read****.*
- 0 points if lacks naming conventions, does not use indentation and white space, is written inconsistently. ***Code is difficult to read*****.**

Use of Comments (10 points)

- 10 points if comments are used where appropriate, the program includes a comment block at the beginning stating the purpose of the program, along with its inputs and outputs, all comments reflect the reality of the code and the code contains no superfluous comments.
- 5 points if occasional comments are inaccurate, unnecessary, or necessary but missing, or the beginning comment block lacks some key details.
- 0 points if comments are not used, the comment block at the beginning is missing or lacks major ideas or the program contains many comments that are wildly inaccurate, misleading, or superfluous.

Difficulty level of the game (20 points)

- 20 points if the game is reasonably complex
- 10 points if the game is simple