Date handed out: 26 March 2018, Monday
Date submission due: 9 April 2018, Monday, 23:55

Programming Assignment 2: Zonk

Purpose:

The main purpose of this programming assignment is to revise the topics that we have covered in the first six weeks including fundamentals of C programming, conditional statements, repetitive statements, and functions.

Description:

You will write a program for playing a simplified version of the Zonk game between a player and computer. Zonk is a dice game which was introduced in 1970s. This will give you practice with all three control constructs (sequence, selection and repetition). We are including some design constraints in the "programming task" section, so you will also use functions. This will give you the experience of decomposing a problem into parts, and then implementing each part in one highly cohesive, loosely coupled function.

Don't try to compile your entire program in one "big bang." Compile it piece by piece. Test each piece that you have compiled to make sure it works correctly before you add the next piece.

Zonk Rules:

Equipment: 6 dices and a scoresheet
Number of Players: 2 players
How to play: At the beginning to decide who is going to start the game, each player rolls one die. Highest roll goes first and play proceeds with the other player. The first player rolls all six dice to start his turn. Certain number combinations are worth game points. The player must play at least *one* scoring combination each time he rolls by setting the scoring die (or dice) to the side. He then picks up the remaining dice and rolls them. Again, at least one scoring combination must be played and set aside, and the remaining dice rolled. This sequence continues until the player can stop and keep his points for the turn, or **zonks**.
Zonking: If, after any roll (the number of die or dice depends on users decision), no scoring combinations turn up, the player zonks and his turn ends. He scores no points for the turn
Keeping Points: A player may stop his turn and keep all points accumulated during the roll if the following condition is met: The player has at least 350 points in this turn. Otherwise the player has to roll again.
Scoring: The following dice combinations are worth points:

1. Each 5 is worth 50 points.
2. Each 1 worth 100 points.
3. Any *three-of-a-kind* rolled at once is worth *100x* the face value of a single die. (e.g., three two's are worth 200, three three's are worth 300, three four's are worth 400, three five's are worth 500 and three six's are worth 600), apart from three one's which worth 1000 (Note: they don't have to be consecutive).
4. Six of a kind (e.g. 4 4 4 4 4 4 , 5,5,5,5,5,5,,…) are worth 1000 points.
5. A straight (1,2,3,4,5,6) is worth 1500 points (they do not need to be in that order, for example, it could be 3, 2, 1, 6, 5, 4).

When dices are rolled, only one of the above can be used for scoring. Depending on the number of dices kept, one of the strategies will be for scoring. For example, if the user rolls "1 1 1 1 1 1" and they decide to keep 1 of those, and roll again the rest, they will get 100 points. If they decide to keep 3 of those and roll again the other three, they will get 1000 points. If they decide to keep all, then they will get 1000 points. Please see "***computer_strategy_decider***" function definition below for how computer will decide which strategy to use.

Scoresheet: A Scoresheet looks as follows:

Player 1	Player 2
Z	Z
600	Z
1150	Z

As you can see scores are accumalted from the previous round.

How to Play Zonk?

You will write the program that will allow a player play the Zonk game against the computer. Your program will allow the players play the game for as many rounds as they specify at the beginning of the game and who ever has the highest score at the end, will win the game. Your game will start with rolling the dice for the computer player and then asking the other player if they are ready to play, a sample run is as follows:

Welcome to the Zonk game.
How many rounds would you like to play? 2
OK, lets get started!

I have rolled the dice and got 3!
Are you ready to Zonk! Shall I roll the dice for you (Y/N)? Y
I have rolled the dice for you and you got 1!

My Turn:

I got → [Dice 1]: 1 [Dice 2]: 2 [Dice 3]: 3 [Dice 4]:4 [Dice 5]:4 [Dice 6]: 4
Kept dice 4 5 6 and Rolled 1 2 3
I got → [Dice 1]: 2 [Dice 2]: 2 [Dice 3]: 2
I stop and kept dice 1 2 3 4 5 6
My score: 600

Your Turn:

Are you ready to Zonk! Shall I roll them for you (Y/N)? Y
You got → [Dice 1]:1 [Dice 2]: 1 [Dice 3]:1 [Dice 4]:3 [Dice 5]: 2 [Dice 6]: 2Do you want to continue (Y/N)? Y
Which ones you want to roll again? 4 5 6
You got → [Dice 4]: 1 [Dice 5]: 1 [Dice 6]: 1
Do you want to continue (Y/N)? N
Your score: 2000

Our scoresheet:
==========
My score Your score
600 2000

My Turn:

I got → [Dice 1]: 2 [Dice 2]: 2 [Dice 3]: 3 [Dice 4]:4 [Dice 5]:4 [Dice 6]: 4
Kept Dice 4 5 6 and rolled 1 2 3
I got → [Dice 1]: 2 [Dice 2]: 3 [Dice 3]: 3
ZONK!!!!
My score: Z

Your Turn:

Are you ready to Zonk! Shall I roll them for you? (Y/N)
You got → [Dice 1]:4 [Dice 2]: 4 [Dice 3]:4 [Dice 4]:3 [Dice 5]: 2 [Dice 6]: 2
Do you want to continue (Y/N)? Y
Which ones you want to roll again? 4 5 6
You got → [Dice 4]:1 [Dice 5]: 4 [Dice 6]:4
Do you want to continue (Y/N)? Y
Which ones you want to roll again? 5 6
You got → [Dice 5]:5 [Dice 6]: 5

Do you want to continue (Y/N)? N
Your score: 600

Our scoresheet:
==========

My score Your score
600 2600

YOU ARE THE WINNER!

Programming Requirements:

In order to implement this game you will need to write at least the following functions, but if you need more functions you can add them.

roll-a-dice () – This function will roll a dice and return the result. The rolling action will give random values.

play_computer() – This function will mainly be responsible from making the computer play the game.

computer_strategy_decider() – This function will decide which scoring strategy will be used. In summary, the computer will always play safe and have a very simple strategy. The computer will always choose the strategy that will give her the highest score (see scoring section). It will check all the strategies and decide which one will give her the highest score. For example, if the dices look as follows: 1, 2, 3, 4, 5, 6 then there is no need to check other strategies the scoring is 1500. If the scoring is as follows 1,1,1,5,5,5 then the computer will keep the first three and roll the last three as this will give her 1000 scoring point rather than keeping 5,5,5 which would give her 500.

play-user() – This function will manily be used to get the player play a turn.

scoresheet() – This function will be used to display the scoresheet on the screen.

Grading Schema:

Your program will be graded as follows:

Grading Point	**Mark (100)**
Maintaining the number of rounds requested by the user and also maintaining the total cores	10
roll-a-dice() function	10
play_computer() function	20
computer_strategy_decider() function	20
play_user() function	20
scoresheet() function	10
Code quality (e.g., variable names, formulation of selection statements and loops, etc)	10

Rules:

Please make sure that you follow the restrictions for the assignment as follows.

- **Strictly obey the input output format. Do not print extra things.**
- **You are not allowed to use global variables.**
- **You are not allowed to use data structures such as arrays to store values as we have not covered them in the class yet.**
- **Add your name/surname and ID at the top of your code as comments and name your source file "CNG140-P2.c"**
- **Upload only source file. Do not compress it (zip, rar, …)**