MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG140 C Programming – Programming Assignment 2

Date handed out: 21 March 2024, Thursday
Date submission due: 4 April 2024, Thursday, 23:55, Cyprus Time

Programming Assignment 1: Crag

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:

Crag is a dice game, that is played with three dice. You will write a program for playing a Crag game between a player and a computer. 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, for example, you will need to 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 parts as functions.

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.

Crag Game Rules:

Equipment: 3 dice 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. The player who rolls the highest die starts the game. The game consists of N rounds – each player will play N turns. The game is played with 3 dice. The first player rolls all three dice to start his/her turn. Players take turns rolling three dice and assign them to certain combinations according to the scoring table given below. After throwing the dice, a player may choose to reroll any number of those dice. This second roll is final. The player doesn't have to use all two throws and may stop after the first. The scoring categories have varying point values, some of which are fixed values and others where the score depends on the value of the dice. The winner is the player who scores the most points.

The computer will only choose to roll the dice again if it is very close to doing high straight, but the other player needs to be able to choose to re-roll the dice or not. Note that the player can roll at most two times.

Scoresheet: A Scoresheet looks as follows:

Computer	Player 2
8	28
28	43

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

Scoring:
You will apply the rules listed in the table below from top to bottom. This means if dice values satisfy more than one category then the one with the highest score will always be used. If point calculations involves order, then the dice values will be checked in order. For example, in order to have low straight points, then the first die should be 1, the second one will be 2 and the third one should be 3, in that order.

Below you can see the scoring table:

Category	Description	Score	Example
Crag	Any combination containing a pair and totalling 13	50	Dice shows: 6 6 1
Thirteen	Any combination totalling 13	26	Dice shows: 3 4 6
Three-Of-A-Kind	Three dice showing the same face	25	Dice shows: 4 4 4
Low Straight	1-2-3	20	Dice shows: 1 2 3
High Straight	4-5-6	20	Dice shows: 4 5 6
Odd Straight	1-3-5	20	Dice shows: 1 3 5
Even Straight	2-4-6	20	Dice shows: 2 4 6
Sum	Any other combination	Sum of all dice values	Dice shows: 3 2 6 (score will be 11)

How to Play Crag?

You will write the program that will allow a player to play the Crag game against the computer. The game consists of N rounds. At the end of N rounds, whoever has the highest score will win the game. The user will choose the number of rounds, N.

A sample run is as follows where "My Turn" represents a computer!

```
Welcome to Crag game.
Let's get started!

How many rounds would you like to play? 3

I have rolled the dice and got 3!
I have rolled the dice for you and you got 6!

Round 1 -- Your Turn:
```

```
You got → [Dice 1]: 1 [Dice 2]: 2 [Dice 3]: 2

Shall I roll for you (Y/N)? Y
Which ones do you want to keep?  4 5
Sorry, wrong input!

Which ones do you want to keep?  2 3

You got → [Dice 1]: 5 [Dice 2]: 2 [Dice 3]: 2

Your score: 9 Total score: 9


Round 1 -- My Turn:
```

```
I rolled them and I got
→ [Dice 1]: 4 [Dice 2]: 5 [Dice 3]: 1
Rolled dice 3!
→ [Dice 1]: 4 [Dice 2]: 5 [Dice 3]: 6

My score: 20 Total score: 20
```

```
Our scoresheet after round 1:
============================
My score    Your score
20          9

Round 2 -- Your Turn:
```

```
You got → [Dice 1]: 1 [Dice 2]: 3 [Dice 3]: 5
Shall I roll for you (Y/N)? X
Sorry, I don't understand!
Shall I roll for you (Y/N)? N

Your score: 20 Total score: 29


Round 2 -- My Turn:
```

```
I rolled them and got
→ [Dice 1]: 2 [Dice 2]: 4 [Dice 3]: 6
My score: 20 Total score: 40


Our scoresheet after round 2:
============================

My score    Your score
40          29


Round 3 -- Your Turn:
```

```
You got → [Dice 1]: 6 [Dice 2]: 6 [Dice 3]: 2
Shall I roll for you (Y/N)? Y
Which ones do you want to keep? 1 2

You got → [Dice 1]: 6 [Dice 2]: 6 [Dice 3]: 1

Crag!!

Your score: 50 Total score: 79


Round 3 -- My Turn:
```

```
I rolled them and got
→ [Dice 1]: 1 [Dice 2]: 3 [Dice 3]: 5
My score: 20 Total score: 60


Our scoresheet after round 3:
============================

My score    Your score
60          79

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 a random value representing a possible dice value.

play_computer() – This function will mainly be responsible from making the computer play the game. The computer will only choose to roll dices again if it is very close to doing high straight (4-5-6 in this order). **Some** example cases are as follows (Please note that the following list does not include all possible combinations):

If the computer rolls in this order: 4,5,5 then the computer will roll dice 3 so that it can be 4,5,6.

If the computer rolls in this order : 4,5,1 then the computer will roll dice 3 so that it can be 4,5,6.

If the computer rolls in this order: 4,5,2 then the computer will roll dice 3 so that it can be 4,5,6.

If the computer rolls in this order: 3,5,6 then the computer will roll dice 1 so that it can be 4,5,6.

If the computer rolls in this order: 4,3,6 then the computer will roll dice 2 so that it can be 4,5,6.

If the computer rolls in this order: 1,5,6 then the computer will roll dice 1 so that it can be 4,5,6.

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

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

Grading Schema:

If your code does NOT compile, you will automatically get zero. If your code compiles, you will then be graded based the following scheme:

Grading Point	**Mark (100)**
Maintaining the number of rounds requested by the user and also maintaining the total scores	10
roll-a-dice() function	10
play_computer() function	25
play_user() function	35
scoresheet() function	10
Code quality (e.g., formatting, commenting, naming variables, clean use of C constructs such as formulation of selection statements and loops, etc)[1]	10

If you do not obey the following rules then you will automatically get 0.

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 and goto statements.**
- **You are NOT allowed to use data structures such as arrays/strings to store values as we have not covered them in the class yet.**
- **If you use break/continue statements in your loops, you will be penalized. Therefore, please avoid using break/continue in the formulation of your iterations.**
- **You need to organize your code in the given functions, however if you do not have functions, then your grade will be out of 50%.**

1 See guidelines given here: https://www.gnu.org/prep/standards/html_node/Writing-C.html

- **Add your name/surname and ID at the top of your code as comments and name your source file "main.c".**
- **Submit your "main.c" to odtuclass. Do not compress your project or submit other files (zip, rar, …). You only need to submit "main.c".**

Cheating:

Full description of cheating, please refer to https://ncc.metu.edu.tr/res/academic-code-of-ethics, but here we provide a description for assignments in this course. The following will be considered as cheating and the Professionalism and Ethics rules given in the course syallbus will be applied accordingly:

1. Students are expected to complete the assignments on their own. Working with other students in assignments teamwork is NOT allowed (Working together is limited to brainstorming only. The "effort" in the assignments should belong completely to the student);
2. Submitting assignments, taken partially or entirely from others or from a source (book, internet, paper, etc);
3. Submitting assignment/exam, taken partially or entirely from a previously graded work;
4. Using generative AI tools to generate solutions and/or presenting generative AI solutions as one's original work will also be considered cheating.