COMP1917 Computing 1

Session 2, 2016

Assignment 1 - Simplified Yahtzee

Due:
Marks: 10% of final assessment

1 Preliminary

For this assignment, you'll implement a simplified scoring system for Yahtzee.

1.1 What is Yahtzee?

Yahtzee is a dice game. You roll five dice and get points based on poker-themed combinations[1]:

- **Three of a Kind** score: sum of all the dice if at least three dice show the same number.
- **Four of a Kind** score: the sum of all the dice if at least four dice show the same number.
- **Full House** score: 25 points if two of the dice show the same number and the rest show another number, i.e., a pair and a triple.
- **Small Straight** score: 30 points if four dice show consecutive numbers.
- **Straight** score: 40 points if the dice show five consecutive numbers.
- **Yahtzee** score: 50 points if all dice show the same number.
- **Chance** score: the sum of all the dice (no condition).

Each player has a table with one entry for each combination. After throwing the dice, the player decides which combination he or she wants to score. For example, if three of the dice have the value 4, two the value 3, it could be counted as Three of a Kind, Full House, or Chance. The player has to pick one of the possible combinations and enter the score in the appropriate field in the table, but only if this entry is still empty. Alternatively, the player can decide to enter 0 in any of the empty entries of the table,

The game continues until every entry of the score table is filled. The maximum score that you can get from a single game is 1575. Yahtzee can be played by multiple players to beat each other, but it can also be played by a single player to reach the maximum score.

The assignment is to implement a subset of Yahtzee.

1.2 Where can we learn more about Yahtzee?

- `http://en.wikipedia.org/wiki/Yahtzee` has well-organised information about the rules and the history of the game.
- `http://www.rekenwonder.com/yahtzee/yahtzee.htm` is where you can play Triple Yahtzee, which is a variation of the standard Yahtzee.

[1]There are more possible combinations, but for the sake of simplicity, we'll ignore them for this assignment

2 Stage 1: Dice Valuation

For the first stage of the assignment, you'll implement the valuation.

2.1 Dice Values Input

Read the five dice values from the standart input, separated by whitespaces[2].

Error Checking You need to check if the player enters the correct inputs. Otherwise, your program needs to print the following error messages and exit:

- **Incorrect Input Format** when the values are not separated by whitespaces.

  ```
  Please enter dice values:
  > 3,4-3 6 5 2

  Incorrect Input Format.
  ```

- **Value Out of Range** when some of the values are greater than 6 or smaller than 1,

  ```
  Please enter dice values:
  > 7 3 4 1 5

  Value Out of Range.
  ```

- **Incorrect Number of Values** when the number of the values that the player entered is not equal to 5

  ```
  Please enter dice values:
  > 3 4 3 6 5 2

  Incorrect Number of Values.
  ```

You have to check for errors in the order given above. That is, if the values that the player entered contain more than one error conditions, the error message with the highest priority has to printed:

```
$ ./assign1

Please enter dice values:
> 3 7 2 2 1 5

Value Out of Range.

$ ./assign1

Please enter dice values:
> 3.4, 7, 6, 2, 1

Incorrect Input Format.
```

[2]We'll see later how we can get random values for the dice. For test purposes, we'll just read the values.

It is important to stick to these rules, and use exactly the error messages listed in the specification. We use automarking to test your program, so even small discrepancies can cause the autotest to fail. We will give you sample test files, so you can test your implementation against the specification.

Once the program has accepted valid input values, it asks the player whether he or she wants to reroll dice, giving three options: *Reroll some dice*, *Reroll all dice*, and *Keep dice.*

For the first stage, simply ignore the first two cases and exit, and proceed only with the last option.

```
Please choose:
 1 -> Reroll some dice
 2 -> Reroll all dice
 3 -> Keep dice
> 3
```

If the player chooses an invalid option, *Invalid Choice* error message must be displayed and your program must terminate. The choice is invalid when it is not one of the three options given. If the user enters more than one choice, your program takes the first one and ignores the rest.

```
Please choose:
 1 -> Reroll some dice
 2 -> Reroll all dice
 3 -> Keep dice
> 2.5

Invalid Choice.
```

2.2 Valuation

To valuate the dice, check for *Three of a Kind*, *Four of a Kind*, *Full House*, *Small Straight*, *Straight*, *Yahtzee*, and *Chance.*

All the possible combinations with the points must be printed out as the score options, so that the player can choose one of them. The score options must be printed in order of *Three of a Kind*, *Four of a Kind*, *Full House*, *Small Straight*, *Straight*, *Yahtzee*, and *Chance.* For example, if the dice values are 1, 2, 3, 4, and 5, the order of the score options that your program displays has to be *Small Straight*, *Straight*, and *Chance.*

At the end, your program must display the score that the player has gained.

```
Your score options are:
 1 -> Small Straight (30 points)
 2 -> Straight (40 points)
 3 -> Chance (15 points)
> 2

Your score is:  40 points
```

If the player chooses an invalid option, *Invalid Choice* error message must be displayed and your program must terminate. The choice is invalid when it is not one of the three options given. If the user enters more than one choice, your program takes the first one and ignores the rest.

```
Your score options are:
 1 -> Small Straight (30 points)
 2 -> Straight (40 points)
 3 -> Chance (15 points)
> 5

Invalid Choice.
```

2.3 Sample Output

After this stage, your program must run as shown below:

```
$ ./assign1

Please enter dice values:
> 6 6 6 4 4

Your dice values are: 6 6 6 4 4

Please choose:
 1 -> Reroll some dice
 2 -> Reroll all dice
 3 -> Keep dice
> 3

Your score options are:
 1 -> Three of a Kind (26 points)
 2 -> Full House (25 points)
 3 -> Chance (26 points)
> 2

Your score is: 25 points

$
```

3 Stage 2: Dice Rerolling

For the second stage, modify your program so that the player has two chances to reroll all or just a selection of the dice.

3.1 Reroll some dice

The player gets to choose which dice to reroll by entering dice numbers. You are, again, required to check the validity of the values entered. Otherwise, your program needs to print out the appropriate error message and exit:

- **Incorrect Input Format** when the values are not separated by whitespaces.

```
Please enter dice values:
> 3, 6

Incorrect Input Format.
```

- **Dice Number Out of Range** when some of the dice numbers are greater than 5 or smaller than 1,

```
Please enter dice to reroll (1 - 5):
> 3 5 8
Dice Number Out of Range.
```

Again, make sure that error checking works in the above order. That is, if the values that the player entered contain more than one error condition the error message with the highest priority is printed out.

If one dice number occurs more than once, they are considered as a single input. For example, if the input is 3, 3, and 5 for the dice numbers, your program must recognise the input as 3 and 5.

Your program displays the number of the dice values to be entered by the player and accepts new dice values for the dice rerolled.

```
Please enter dice to reroll (1 - 5):
> 3

Please enter 1 value:
> 3
```

```
Please enter dice to reroll (1 - 5):
> 2 4

Please enter 2 values:
> 5 6
```

New dice numbers need to be checked with the error conditions in Section 2.1.

3.2 Reroll all dice

Your program must accept five dice values in the same way that it accepts the initial dice values. Those values go through the error checking process as shown in the Section 2.1.

3.3 Sample Output

After this stage, your program must run as shown below.

```
$ ./assign1

Please enter dice values:
> 1 3 3 4 6

Your dice values are: 1 3 3 4 6
```

```
Please choose:
 1 -> Reroll some dice
 2 -> Reroll all dice
 3 -> Keep dice
> 2

Please enter dice values:
> 1 3 1 4 5

Your dice values are: 1 3 1 4 5

Pleaes choose:
 1 -> Reroll some dice
 2 -> Reroll all dice
 3 -> Keep dice
> 1

Please enter dice to reroll (1 - 5):
> 1

Please enter one die value:
> 2

Your dice values are: 2 3 1 4 5

Your score options are:
 1 -> Small Straight (30 points)
 2 -> Straight (40 points)
 3 -> Chance (15 points)
> 2

Your score is: 40 points

$
```

4 Hints

4.1 Getting Started

Don't try and write the program straight away. First, make sure that you understand each part of Stage 1 of the assignment. Simulate the behaviour of the program on a piece of paper. Write down the individual steps necessary to implement Stage 1. Try and break each step down into simple substeps as far as possible.

To implement the assignment, you have to know about arrays and functions, which we will cover in Week 4 in the lecture.

4.2 Reading Input Values

You already know how to use `scanf` and `getchar` to read from the keyboard. To check the format of the input that the player enters, we recommend to use `getchar`. It'll simplify the error checking processes. For instance, with `scanf`, you cannot properly check whether the player used whitespaces as the delimiters.

Once you have read inputs using `getchar`, you need to convert the character to the corresponding number, i.e., the character `‘1’` to the integer value `1`. In C, every character has its own numeric value (its ASCII code), and arithmetic operations, such as addition, subtraction, multiplication, and division, are applicable to them. The character, `‘1’`, has the numeric value 49, and `‘1’ - 48` yields you the numeric value 1. In the same way, `‘2’ - 48` yields the numeric value 2. For more information about the numeric values for the corresponding characters can be found at `http://www.asciitable.com`, and using `man ascii` command.

As a first implementation step, write a program that reads the dice values, checks if they are in the required format. If they are, it should simply print the values, otherwise print the specified error messages.

4.3 Data Structure

You need to maintain the information about the dice. For the assignment, you have to be able to store the value of a die, and calculate the score. For the second stage of the assignment, you also have to be able to update the value of a die — that is, delete its old value and enter the new value.

The straight forward solution would be to have just an array with one entry for each die (i.e., 5), such that each entry contains the value of that die. Take a piece of paper to see how you can check for the different combinations using this representation. You’ll probably find that it is fairly complicated.

Another possibility is to have an array with six entries, one for each possible value of a die: the first entry contains the number of times 1 has been scored, the second the number of times 2 has been scored, and so on. Now, try again to check for different combinations on this representation. This time, it is much easier. For Stage 2, however, you’ll find that the frequency representation is not sufficient. If the player wants to reroll, say, die number 2, you don’t know anymore what it’s value was.

5 Submission

You must submit one ANSI C source file which must be called assign1.c which, when compiled with `-Wall` and `-Werror`, will produce an executable that performs exactly as described in the specification. Once submissions are open, you should submit by typing

```
give cs1917 assign1 assign1.c
```

You can submit as many times as you like - later submissions will overwrite earlier ones. You can check that your submission has been received by using the following command:

```
1917 classrun -check
```

The submission deadline is Sunday 4th September 23:59:59. If you submit the assignment after the deadline, you will lose 10% of the maximum marks for each day you are late. You’ll get 0 marks if you submit more than five days late.

Additional information may be found in the FAQ and will be considered as part of the specification for the assignment. Questions relating to this assignment can also be posted to the comments section of the assignment 1 page of the course website.

If you have a question that has not already been answered in the FAQ or in the comments, you can email it to your tutor, or to `s.mautner@unsw.edu.au`.

6 Marking Scheme

Assignment 1 is worth 10 marks (which will make up 10% of the overall marks for the course). Stage 1: 5 marks Stage 2: 3 marks Style: 2 marks