***Under 40,*** Dice Game –

* This game is of my own creation. Since it does not exist anywhere else, it is not published online and hence students cannot “borrow” or ”steal” code from online.
* If you use this or a variation of, please do not publish or post solutions accessible over the public Internet
* Game was designed with the Gr 11 ICS 3U/C class in mind and is scaffolded to both demonstrate various programming skills (decisions, loops, methods etc) and encourage the breakdown of a problem into various smaller and manageable pieces (ICS 3C class does not complete the arrays section)
* Coding can be done independently or started independently and then integrated with partners/groups at the methods stage by using the designs and ideas of multiple people
* There is a checklist included for both student/teacher use if desired
* Many adaptations of the game can be derived to make unique between classes (scoring changes, number of dice, number of faces on dice, end goal (40,1000), how players take turns (i.e. each player rolls in succession until last player standing), etc.)

***Under 40,*** dice game

Object:  is to get a score as close to the score of 40 without going over in each round

Rules:

1. Each player roll 2 dice to determine who starts. Player with the highest roll goes first. High ties reroll.
2. Starting at a score of 0 for each round, a player rolls both dice and adds a score based on the following rules:
   1. If doubles are rolled, score is double the result of multiplying the pips.
   2. If there is one or more even dice, score is the result of multiplying pips.
   3. If both dice are odd, score is the result of multiplying the pips and subtracting the lower of the two.  
      (Note: you will always be adding an even number to your score.)
3. A player’s turn includes rolling both dice, identifying the new cumulative score and then decides if they can/want to roll again to get closer to 40.
   1. If the player scores 40 exactly, the round ends and they earn 40 times the number of players as points. Next round starts with the player on the left (clockwise).
   2. The starting player in the round must roll once and may choose to stop at any time as long as there score is under 40.
   3. If a players score goes above 40, they bust and their score will be counted to be awarded to the winner of the round.
4. Once the starting player stops, play passes to the left and they roll until they get a score higher than the previous player and closer to 40 or bust.
5. Round ends when one player reaches exactly a score of 40 or everyone but one person busts.
6. Next round begins with player to the “left” of the initial starting player.
7. Winner of the round accumulates points at the end of the round by the sum of all the busted player scores or if they earn exactly 40, they earn 40 times the number of players in the game for the round.
8. First player who reaches 1000 points wins.

ICS3U/C programming task:

You will program this game in stages and these stages should be done in order. You must complete a minimum of 3 stages in each unit to meet the expectations for this project.

U3 Decisions:

1. Player goes first. Create two random numbers to represent the dice. Calculate the current score. Print dice rolls and the score to the screen.
2. Computer’s turn second. Create two random numbers to represent the dice. Calculate the current score. Print the dice rolls and the score to the screen.
3. Compare Player and Computer score to see who won (closest to 40 without going over).
4. Program who goes first (see #1)
5. Ask for the user’s name and use this name in the dialogue with the user.
6. Sentinel to quit the game

U4 Loops:

1. U3 Decisions stages (first 3 minimum)
2. If valid, ask if player wants to roll again until they say no, or bust.
3. If valid, decide if computer should roll again, or bust
4. Different number of players? Including or not including computer as a player?
5. Keep track of the scores/points for 3 rounds to determine winner OR
6. Ask for number of rounds player wants and keep track of the scores/points to determine winner.

U5 Methods:

1. U4 Loops stages (first 3 minimum)
2. Make methods for
   * 1. Instructions - void
     2. Roll Dice or PlayerTurn - function
     3. Add/Find Score - parameter (or function)
3. Make methods for at least one other reusable section of your code
4. Make error checking methods that not only checks for appropriate ranges/values but also checks for appropriate type of data entered  (Try .. Catch, method overloading)

U6 Arrays:

1. U5 Methods stages (first 3 minimum)
2. Create and use at least two arrays to store/manipulate data

I.e - Player Names, Player Scores, Current Player Information etc.

Checklist:      How Well Did You …. ?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Planning / Efficiencies: | | | |
|  | Planning Submitted? | | |  |
|  | Group/Individual Elements To Be Incorporated Are Identified (with justification) | | |  |
|  | Creative Elements | | |  |
|  |  | Above and Beyond | |  |
|  | Code Efficiencies | | |  |
|  |  | Methods (reduce repetative code, chunked appropriately) | |  |
|  |  | Appropriate / efficient algorithms | |  |
|  | Communication: | | | |
|  | Readability for User | | |  |
|  |  | Spelling | |  |
|  |  | Formatting on Screen | |  |
|  |  | Timing (if applicable) | |  |
|  |  | Instructions/directions clear | |  |
|  | Readability for Programmer | | |  |
|  |  | Variable and Method Naming | |  |
|  |  | Camel Case | |  |
|  |  |  | ClassNames |  |
|  |  |  | methodNames |  |
|  |  |  | variableNames |  |
|  |  | Indending | |  |
|  |  | Alignment of brackets | |  |
|  |  | Spacing | |  |
|  |  | Chunking of code | |  |
|  |  | Commenting | |  |
|  |  |  | "Above" chunks of code |  |
|  |  |  | Beside code for elaboration only |  |
|  |  |  | Methods have appropriate commenting information |  |
|  |  |  | Classes have appropriate commenting information |  |
|  |  |  | End brackets commented |  |
|  |  |  | Internal comments effective (appropriately descriptive and concise) |  |

Checklist:      How Well Did You …. ?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | U3 Decisions: | | | |
|  | Player goes first. | | |  |
|  |  | Create two random numbers to represent the dice. | |  |
|  |  | Calculate the current score. | |  |
|  |  | Print dice rolls and the score to the screen. | |  |
|  | Computer’s turn second. | | |  |
|  |  | Create two random numbers to represent the dice. | |  |
|  |  | Calculate the current score. | |  |
|  |  | Print dice rolls and the score to the screen. | |  |
|  | Compare Player and Computer score to see who won (closest to 40 without going over). | | |  |
|  | Optional |  |  |  |
|  |  | Program who goes first (see #1) | |  |
|  |  | Ask for the user’s name and use this name in the dialogue with the user. | |  |
|  |  | Sentinel to quit the game | |  |
|  |  |  |  |  |
|  | U4 Loops: | | | |
|  | Roll Again |  |  |  |
|  |  | Player | Bust |  |
|  |  |  | =40 |  |
|  |  |  | Say Stop (if valid) |  |
|  |  | Computer | Bust |  |
|  |  |  | =40 |  |
|  |  |  | Say Stop (if valid) |  |
|  | Optional |  |  |  |
|  |  | Different number of players? Including or not including computer as a player? | |  |
|  |  | Keep track of the scores/points for 3 rounds to determine winner OR | |  |
|  |  | Ask for number of rounds player wants tracks scores/points to determine winner. | |  |
|  |  |  |  |  |
|  | U5: Methods | | | |
|  | Void | Instructions | |  |
|  | Function | Roll Dice or PlayerTurn | |  |
|  | Parameter | Add/Find Score (or function) | |  |
|  | Make methods for at least one other reusable section of your code | | |  |
|  | Optional |  |  |  |
|  |  | Make error checking methods | |  |
|  |  |  | ranges/values |  |
|  |  |  | data type (int vs String) |  |
|  |  |  | Try ... Catch |  |
|  |  |  | Method Overloading |  |
|  | U6: Arrays | | | |
|  | Create and use at least two arrays to store/manipulate data | | |  |
|  |  | Player Name | |  |
|  |  | Player Score | |  |
|  |  | Current Player | |  |
|  |  | Other? | |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Planning / Efficiencies / Testing: | | | |
|  | Testing |  |  |  |
|  |  | Testing documented | |  |
|  |  |  | Base Cases - all valid inputs |  |
|  |  |  | Ranges |  |
|  |  |  | Invalid Inputs |  |
|  |  |  | Methods |  |
|  |  | Recognized issues and developed solutions (i.e. 1st player 2nd round etc.) | |  |