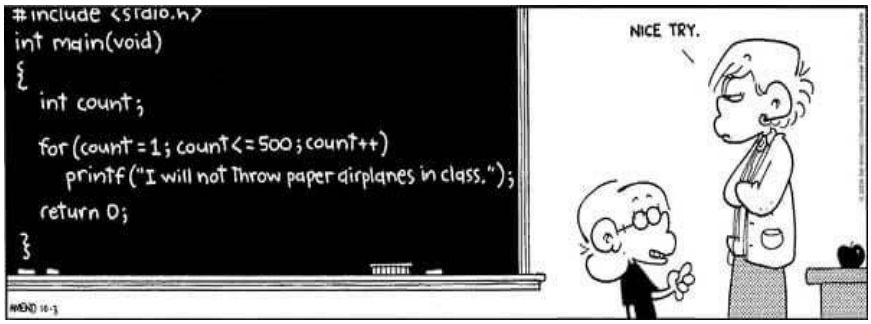
CS 202 – Assignment #2

Purpose: Learn simple class implementation.
Points: 100

Video Link for Assignment2:
https://unlv.webex.com/unlv/ldr.php?RCID=ed2c9501d8bb35da9b8cf7e5da1dead7

Assignment:



Design a C++ class to simulate and play simplified game of craps[1] game. Specifically, you should create an object that will provide the functionality for the simulated, simplified game of craps based on the provided rules. *Note*, the simplifications are primarily related to the limiting of the betting options.

The provided main(int argc, char ** argv) function will serve as a driver for the program which uses the craps game object. The *crapsType* UML class diagram is as follows:

crapsType
-originalStake: int
-cashOnHand: int
-display: bool
-MINSTAKE = 50: static const int
-MAXSTAKE = 10000: static const int
-MINBET = 10: static const int
-MAXBET = 1000: static const int
+crapsType()
+readStake(): void
+getStake() const: int
+setStake(int): void
+getCurrentBalance() const: int
+setDisplay(bool): void
+playCraps(int): bool
-displayRoll(int, int): void

You will need to develop an implementation file based on the above UML diagram. Your implementation file should be fully commented. Refer to the example executions for output formatting.

Due to the simplified rules, this is not likely to be useful in developing a winning strategy for playing the real game.

1 For more information, refer to: http://en.wikipedia.org/wiki/Craps

Linking Instructions:
A **make** file is provided. As projects get larger, compiling a series of source files by hand becomes tedious. Make is a utility that will read a make file (named **makefile** by default), compile applicable source(s), and build the executable file. Assuming the main file is named **`main.cpp`**, and the implementation file is named **`crapsTypeImp.cpp`**, the following are the instructions to build the executable.

```
kishore-vm:kishore$ make
```

Which will create an executable named **main**. The provided make file assumes these file names. As such, you should not change the file names for this assignment.

To run the code, the user have to pass two arguments:

```
kishore-vm:kishore$ ./main 7
```

From above, the second argument(int) is for the seed value. When you execute your code, every time you will get the same output as it has the same seed value. The CodeGrade can verify your output by comparing with sample execution.

Program Header Block
All program and header files must include your name, section number, assignment, NSHE number, input and output summary. The required format is as follows:

```
/*
Name: MY_NAME, NSHE, CLASS-SECTION, ASSIGNMENT
Description: <per assignment>
Input: <per assignment>
Output: <per assignment>
*/
```

Failure to include your name in this format will result in a loss of up to 5%.

Code Quality Checks
A C++ linter[2] is used to perform some basic checks on code quality. These checks include, but are not limited to, the following:

- Unnecessary 'else' statements should not be used.
- Identifier naming style should be either camelCase or snake_case (consistently chosen).
- Named constants should be used in place of literals.
- Correct indentation should be used.
- Redundant return/continue statements should not be used.
- Selection conditions should be in the simplest form possible.
- Function prototypes should be free of top level *const*.

Not all of these items will apply to every program. Failure to to address these guidelines will result in a loss of up to 5%.

2 For more information, refer to: https://en.wikipedia.org/wiki/Lint_(software)

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Compilation	-	Failure to compile will result in a score of 0.
Program Header	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Line Length	5%	No lines should exceed more than eighty (80) characters.
Code Quality	5%	Must meet some basic code quality checks (see above)
Program Functionality (and on-time)	75%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Craps Rules

The simplified craps rules as as follows:

- Each player rolls two standard, six-sided dice.
- The two dice are added
- If the sum is 7 or 11 on the first throw, it is a "win"
- If the sum is 2, 3, or 12 (called "craps") on the first throw, it is a loss
- If the sum is 4, 5, 6, 8, 9, or 10 that becomes the players "point". To win, the player continues to roll dice until he/she "makes point" (i.e., rolls the same point value). If the player losses if a 7 is thrown.
- If the game is displayed, each roll (die1, die2, sum, and nickname) should be displayed. If the games is not won on the first roll, the point value should be displayed.
- The nicknames should be based on the following table:

	1	2	3	4	5	6
1	Snake Eyes	Ace Deuce	Easy Four	Fever Five	Easy Six	Seven Out
2	Ace Deuce	Hard Four	Fever Five	Easy Six	Seven Out	Easy Eight
3	Easy Four	Fever Five	Hard Six	Seven Out	Easy Eight	Nina
4	Fever Five	Easy Six	Seven Out	Hard Eight	Nina	Easy Ten
5	Easy Six	Seven Out	Easy Eight	Nina	Hard Ten	Yo-leven
6	Seven Out	Easy Eight	Nina	Easy Ten	Yo-leven	Boxcars

Function Descriptions:

The following are more detailed descriptions of the required functions.

- The *crapsType()* constructor should initialize the class variables (integers to zero and boolean to true).
- The *readStake()* function should read the stake amount from the user and verify that it is between MINSTAKE and MAXSTAKE (inclusive). The function should re-prompt for incorrect input (an unlimited number of times). You may assume valid data type entry (i.e., actual numeric data) for the amount. The *originalStake* and *cashOnHand* class variables should be set to the stake amount.
- The *setStake(int)* function should set the stake amount to the passed value and verify that it is between MINSTAKE and MAXSTAKE (inclusive). If the value is valid, the *originalStake* and *cashOnHand variables* should be set. If the value is invalid, an error message should be displayed.
- The *getStake()* function should return the original stake amount.
- The *getCurrentBalance()* function should return the current *cashOnHand*.
- The *setDisplay(bool)* function should set the class variable, display, to the passed value(true or false).
- The *playCraps(int)* function should play one round of craps based on the provided rules. First, the function should verify the bet is between MINBET and MAXBET (inclusive) and, if not, display an error message and return false. If the game is won, the function should return true and *cashOnHand* should be increased by the bet amount. If the game is lost, the function should return false and *cashOnHand* should be decreased by the bet amount.
- The *displayRoll(int die1, int die2)* function should display/print the values of die1, die2, sum(die1+die2), and nickName(check the above table under Craps Rules description). (Hint: You will call this function in *playCraps(int)* function)

Files Provided:

The following files are available to download for this assignment.

- makefile.
- main.cpp.
- crapsType.h

Submission:

- Submit crapsTypeImp.cpp only.

- **Things to Remember**
- All files must compile and execute on Ubuntu and compile with C++11.
- Submit source files
  - Submit a copy of the source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Example Execution:
Below is an example program execution.

```
kishore-vm:kishore$ make
g++ -Wall -Wextra -pedantic -std=c++11 -g -c main.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -c crapsTypeImp.cpp
g++ -Wall -Wextra -pedantic -std=c++11 -g -o main main.o crapsTypeImp.o
kishore-vm:kishore$ ./main 7

---------------------------
CS 202 Assignment #2
Craps Game Simulation.

---------------------------
Game Example A (w/display)
Enter Stake Amount: 3

Error, invalid entry.Must be between 50 and 10000 inclusive.
Please re-enter.
Enter Stake Amount: 100
---------------------------
Die 1: 4     Die 2: 4     Sum: 8 Hard Eight
Point: 8
Die 1: 6     Die 2: 2     Sum: 8 Easy Eight
Won
Current Balance: $200
---------------------------
Die 1: 6     Die 2: 4     Sum: 10      Easy Ten
Point: 10
Die 1: 3     Die 2: 2     Sum: 5 Fever Five
Die 1: 1     Die 2: 2     Sum: 3 Ace Deuce
Die 1: 6     Die 2: 2     Sum: 8 Easy Eight
Die 1: 2     Die 2: 1     Sum: 3 Ace Deuce
Die 1: 2     Die 2: 1     Sum: 3 Ace Deuce
Die 1: 5     Die 2: 3     Sum: 8 Easy Eight
Die 1: 3     Die 2: 1     Sum: 4 Easy Four
Die 1: 3     Die 2: 6     Sum: 9 Nina
Die 1: 1     Die 2: 3     Sum: 4 Easy Four
Die 1: 2     Die 2: 5     Sum: 7 Seven Out

***********************************************
Game Results:
   Game Title: Game Example A (w/display)
   Wins = 1
   Losses = 1
   Rounds = 2
   Original Stake:  $100.00
   Final Balance:  $100.00
   Loser.
***********************************************


Error, Invalid Stake Amount
Error, Invalid Bet

***********************************************
Game Results:
   Game Title: Game Example 1 (wo/display)
   Wins = 43
   Losses = 52
   Rounds = 95
   Original Stake:  $1000.00
   Final Balance:  $100.00
   Loser.
***********************************************

kishore-vm:kishore$
```