ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

## Engineering Programming 1

# Project 1: Blackjack

**Fall 2015**
**Due date: Nov. 13 (23:59 pm)**

*revised: Nov. 02*

# What is Blackjack

- According to Wikipedia (https://en.wikipedia.org/wiki/Blackjack), Blackjack, also known as twenty-one, is the most widely played casino banking game in the world.

- You must implement a console-based Blackjack game using C++ language.

- You will implement much simpler version of the original rules yet you can play the real game online at http://www.247blackjack.com/ for better understanding of the rules.

# Basic Rules

## (Simplified rules for your project)

- The participants of the game are a player and a dealer only.
- Basically the winner is one who gets higher points without exceeding 21 given cards.
- Before starting each game, the player is required to bet for the game. After the bet is made, an initial two cards is given for the player and the dealer each.
  There's no limits about the amount of money the player bets for each game and the player has initial money before starting the first game.
- There's only two decisions the player can made.

  **HIT**

  Take another card from the dealer. If the total score of the player exceeds 21, the player will lose a bet.
  If the player exceeds 21 first, the player loses before dealer.

  **STAND**

  Take no more cards.

# Basic Rules

## (Simplified rules for your project)

- After the player made a decision to STAND**(or HIT)**, the dealer should take another card if the total score of the dealer is less than 17.

- There are no jokers and no patterns in the card set, i.e., the card set only consists of A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K without heart, clover, spade and diamond pattern. **Assume that the number of each card in the card is innumerable.**

- The player decides whether to continue the game whenever each game is ended.

- Note that each player and dealer have only one chance to decide whether to HIT or STAND. That is, unlike the original game rule, after the player gets an initial two cards and made one decision, no chances are allowed to make decision. Hence, the maximum number of cards for both the player and the dealer is 3.

# Basic Rules

## (Simplified rules for your project)

- Score Table

| Card | A | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Score | 1 or 11 | 2 | 3 | 4 | 5 | 6 |

| Card | 7 | 8 | 9 | 10 | J | Q | K |
|---|---|---|---|---|---|---|---|
| Score | 7 | 8 | 9 | 10 | 10 | 10 | 10 |

**A** : 1 or 11 points to the player's advantage.
**2 - 10** : points follows the value on the card surfaces.
**J - K** : 10 points.

# Basic Rules

## (Simplified rules for your project)

- According to the score table, the total score is computed for both the player and the dealer.
- **Assume** betting money is in increments of 10, even number only.
- **Blackjack** is made when the total score of an initial two cards is 21, i.e., when one card is A and the other card is one of 10, J, Q, and K.
- If the player gets total score more than 21, the player loses a bet, since the dealer won't take another card. If the dealer gets total score more than 21, the player wins a bet.
- If both the player and the dealer gets the same total score, the game is pushed(draw), i.e., the game is made void and a new game is restarted if the player wants to.

# Basic Rules

## (Simplified rules for your project)

- The examples of game results:

| Player's score | Dealer's score | Winner |
|---|---|---|
| 20 | 17 | Player |
| Blackjack | 19 | Player |
| 18 | 18 | Draw |
| Blackjack | Blackjack | Draw |
| 22 | 14 | Dealer |
| 18 | 24 | Player |

- The money the player gets or loses is the amount of the bet for each game.
  **If the player wins a bet with Blackjack, the player gets 1.5 times of the bet.**

- It is highly recommended to play Blackjack online if you are not familiar with the original Blackjack rule at http://www.247blackjack.com/

# Main Purpose

- **The main purpose of this project is to make sure that you can write a code with what you've learned from the class:** *basic syntax*, *if-else* statement( or *switch-case* statement), *loop* statement and *rand(), time(),* and *srand() function*.

- **You are not allowed** to use any other stuffs such as *arrays* and *functions*, i.e., whatever you've not learned yet from the lecture slides, to consider fairness between those who are beginners in computer programming and those who are proficient at computer programming. Plus you'd be able to train yourself by this project, handling a logically complicated algorithm.

# Requirements

- **You must read carefully all the requirements.**

- **The requirements can be updated when issues occur.**

# Requirements

- Linux system will be used for evaluating your code. Hence, you are strongly recommended to use Linux system to make sure your code works when evaluated. Note that your code must be compiled in Linux system.

- No libraries are allowed except *iostream, cstdlib, ctime*.
  i.e., no #include statements in your code except:
  #include <iostream>
  #include <cstdlib>
  #include <ctime>

- No function definitions are allowed except the *main() function*.

- You are allowed to use *#define* statement only for defining constants.
  i.e., #define HIT 1
  #define STAND 2

# Requirements

- **Basically, you must implement code which strictly follows all the rules described in the “Basic Rules” section. Also, your code must handle the exceptional cases described in the “Examples” section.**

- You should put comments for explaining what’s happening in your code at least 50 lines using the line comment (starting with //).

- Only *int, char, string,* and *bool* type is allowed. No user-defined data types are allowed.

- The generated cards should be always different whenever your code is executed. Make sure cards are generated randomly.

# Examples

- First, the program will ask the player to enter an initial game money:

```
Initial game money: 10000


>>>>> Game Start!
How much you want to bet? (holding sum $10000) :
```

- The initial game money should be a strictly positive number.

```
Initial game money: -100
Invalid game money. Game money should be more than 0.
Initial game money: 0
Invalid game money. Game money should be more than 0.
Initial game money: 10000


>>>>> Game Start!
How much you want to bet? (holding sum $10000) :
```

# Examples

- After an initial game money is entered, the player should bet some money for the game.

```
>>>>> Game Start!
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 5, Q (sum 15)
Dealer's card deck : 6, ?
Player's choice (1. Hit 2. Stand) :
```

- If the player bet exceeds the holding sum, the program should require the player to enter a bet again.

```
>>>>> Game Start!
How much you want to bet? (holding sum $10000) : 9999999
The money player bet should be less than or equal to player cash($10000)
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 3, 8 (sum 11)
Dealer's card deck : 10, ?
Player's choice (1. Hit 2. Stand) :
```

# Examples

- If the player bet is negative, the program should require the player to enter a bet again.

```
How much you want to bet? (holding sum $10000) : -100
The money player bet should be a positive number
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 4, 4 (sum 8)
Dealer's card deck : A, ?
Player's choice (1. Hit 2. Stand) :
```

# Examples

- When the initial card pair of the player is Blackjack, the winner is determined directly without any player's decision(Revised).

Blackjack (player wins)

```
Initial game money: 10000


>>>>> Game Start!
How much you want to bet? (holding sum $10000) : 100
Player's card deck : A, J (sum 21)
Dealer's card deck : Q, ?
>>>>>>>> BLACK JACK <<<<<<<<
Player win!
Player got $150 (holding sum $10150)

continue game? (y/n) :
```

Blackjack draw

```
Initial game money: 10000


>>>>> Game Start!
How much you want to bet? (holding sum $10000) : 100
Player's card deck : A, J (sum 21)
Dealer's card deck : 10, ?
Dealer's card deck : 10, A (sum 21)
BLACKJACK for both the player and the dealer!(Draw)
Pushing the game...
How much you want to bet? (holding sum $10000) :
```

Normal draw

```
How much you want to bet? (holding sum $1000) :  1
Player's card deck : 10, 9 (sum 19)
Dealer's card deck : 10, ?
Player's choice (1. Hit 2. Stand) : 1
Player's card deck : 10, 9, 2 (sum 21)
Dealer's card deck : 10, ?
>>> Comparing the scores..
Dealer's card deck : 10, 5, 6 (sum 21)
Draw the game. Pushing ...
How much you want to bet? (holding sum $1000) :
```

# Examples

- The player can choose HIT or STAND when given a pair of cards. When a decision is made, the winner for the game is determined.

HIT

```
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 3, 8 (sum 11)
Dealer's card deck : 10, ?
Player's choice (1. Hit 2. Stand) : 1
Player's card deck : 3, 8, 2 (sum 13)
Dealer's card deck : 10, ?
>>> Comparing the scores..
Dealer's card deck : 10, 3, J (sum 24)
Player win!
Player got $100 (holding sum $10100)

continue game? (y/n) :
```

STAND

```
How much you want to bet? (holding sum $10100) : 100
Player's card deck : Q, 6 (sum 16)
Dealer's card deck : 7, ?
Player's choice (1. Hit 2. Stand) : 2
>>> Comparing the scores..
Dealer's card deck : 7, Q (sum 17)
Player lose.
Player lost $100 (holding sum $10000)

continue game? (y/n) :
```

# Examples

- The player can choose HIT or STAND when given a pair of cards. When a decision is made, the winner for the game is determined.

Dealer wins with the score 21 (‘Blackjack’ is only for the player)

```
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 6, K (sum 16)
Dealer's card deck : A, ?
Player's choice (1. Hit 2. Stand) : 2
>>> Comparing the scores..
Dealer's card deck : A, 10 (sum 21)
Player lose.
Player lost $100 (holding sum $9900)

continue game? (y/n) :
```

# Examples

- It is up to the player to bet continuously.

```
Initial game money: 10000


>>>>> Game Start!
How much you want to bet? (holding sum $10000) : 100
Player's card deck : 8, 8 (sum 16)
Dealer's card deck : 6, ?
Player's choice (1. Hit 2. Stand) : 1
Player's card deck : 8, 8, A (sum 17)
Dealer's card deck : 6, ?
>>> Comparing the scores..
Dealer's card deck : 6, 5, 7 (sum 18)
Player lose.
Player lost $100 (holding sum $9900)

continue game? (y/n) : y
---------------------------------------
How much you want to bet? (holding sum $9900) : 100
Player's card deck : Q, 8 (sum 18)
Dealer's card deck : A, ?
Player's choice (1. Hit 2. Stand) : 2
>>> Comparing the scores..
Dealer's card deck : A, J (sum 21)
Player lose.
Player lost $100 (holding sum $9800)

continue game? (y/n) : y
---------------------------------------
How much you want to bet? (holding sum $9800) :
```

# Examples

- When the player does not want to continue games, the program is ended(revised).

```
How much you want to bet? (holding sum $1000) :  1
Player's card deck : 8, 2 (sum 10)
Dealer's card deck : 5, ?
Player's choice (1. Hit 2. Stand) : 1
Player's card deck : 8, 2, 9 (sum 19)
Dealer's card deck : 5, ?
>>> Comparing the scores..
Dealer's card deck : 5, J, Q (sum 25)
Player win!
Player got $1 (holding sum $1001)

continue game? (y/n) : n
----------------------------------------
```

# Examples

- When the holding sum reaches 0, the program is ended.

```
How much you want to bet? (holding sum $100) : 100
Player's card deck : Q, J (sum 20)
Dealer's card deck : 10, ?
Player's choice (1. Hit 2. Stand) : 1
Player's card deck : Q, J, 10 (sum 30)
Dealer's card deck : 10, ?
>>> Comparing the scores..
Dealer's card deck : 10, 7 (sum 17)
Player lose.
Player lost $100 (holding sum $0)

Player lost all of the money


>>>>>>>>>> Game Over!
```

# Examples

- **Note that even though the flow of the program is almost fixed, a prompt format will be given later or soon.**

- Since the prompt format will be only a tiny part of your code, don’t hesitate to start writing your code.

  **Just start to implement your code right now!**

# Grading

- **Grading policy will be updated later or soon.**

# Useful Tips

- Before writing a code, it is recommended to draw a flow chart of the entire game process. If you are not able to draw the flow chart, it's likely you're not aware of how to code.

- Refer to the lecture slide for chapter 3 to remind how to generate a random number.

- Some understanding of ASCII code may be helpful.

# No plagiarism!

- According to our academic dishonesty policy, all projects mustbe  done entirely by yourself, without the help of any other person.

- We will enforce this policy by using a professional-grade program checker to catch students who commit plagiarism. **Students who get caught will receive zero points in the project and/or fail this course immediately.** We  will also inform the university about the incident.

# No plagiarism!

- You must protect your programs from being seen or copied by other people. In case if your program is copied by other students without your acknowledgment, **you will still be considered committing plagiarism.**

- You cannot copy any program that you found on the Internet. Even copying  a small fragment of code is not allowed.

- **If you notice any plagiarism activity, please report to the instructors or TAs.**