CSC2-140 Foundations of Computer Science
Programming Project 3
Due Tuesday, October 24 by 5 p.m. 20 points possible.

You are going to write a blackjack game. Here's a refresher on the rules of Blackjack (a simplified version of Blackjack):
1. The player is dealt two cards.
2. The player takes additional cards one at a time. The player's object is to get as high a score as possible, without exceeding 21 ("busting").
3. Numbered cards count as their natural value. Jack, Queen, and King ("face cards") count as 10.Aces are valued at 1 or 11 according to the player's preference.
4. If the hand value exceeds 21 points, it busts.
The player will play against the dealer, which is the computer. The dealer must continue to take cards until their total is 17 or higher.
If the player busts, he or she loses and the dealer doesn’t even need to play. If the player doesn’t bust, the dealer plays. If the dealer goes bust or scores lower than the player, the player wins. If the dealer scores the same as the player it is a “push”. If the dealer scores higher than the player, the player loses.
If you want to track bets/ runs over a period of time, you are welcome to but this is not a requirement of the program.

Simplifications:
1. For this game, we won't display or represent actual cards, but will in early phases display the total count and in later versions display the card and the suit: 4S could be 4 of spades, etc.
2. We are not going to worry about keeping track of what hand the player already has, just what their total score is.
3. At some tables, the player is asked what value he or she wants for an ace; at others, the dealer chooses the "best" choice for them. Our program should always ask, even if the choice would be obvious (leading to a score of 21, or leading to a "bust").
I recommend that you design this program in phases. In phase 1, don’t worry about duplicate cards or suits and don’t worry about the dealer’s score:
Initially, the game should present the player with a two-card hand: the sum of two random values, each ranging from 1 to 10. Why 1 to 10, and not 1 to 11? Because if the player gets an ace, he or she gets to choose whether or not it should count as 1 or 11. If your program (as the dealer) ever deals a 1, it should ask the user whether or not to add 10 to the total.
Now, as mentioned, the player is dealt two cards, and should be asked if any 1 should be turned into an 11.
The player should then be told his or her score, and so begins a loop:
The player is asked whether to hit or stay. Hit means the program should deal the player one more card; stick means the player is satisfied with that score. If the player types hit, the program should deal another card. The program should continue asking hit or stay until the sum of the card values (including aces valued at 11) hits 21; if it equals 21, the player wins! If, however, it exceeds, 21, the player loses, and should be insulted accordingly!
In phase 2, add suits and tracks duplicate cards.
In phase 3, add the dealer’s score as what they have to beat.
In phase 4, introduce a gui interface to the game.