Today you are creating a simple Black Jack game that simulates the classic game. The user will be able to input a bet amount, then play the game against a computer dealer and make (or lose) fake money!

Overview of the Game Blackjack
For those who have never played Blackjack, here’s a quick overview of the game to get your familiar with what you will be doing.

There are 2 participants – you (the player) and the dealer. The objective of the game is to add cards to your hand until the sum of the cards total 21. You and the dealer can keep adding cards (valued from 1-11) to your hands until you reach 21. Initially you will have two cards, and you can decide if you want the dealer to deal you cards (call “Hit Me”) or to not add any more cards (call “Stay”). If either side goes over 21, they “bust”. When the betting is over and both sides stop adding cards to their hand, the side with the higher value (that MUST be less than or equal to 21). If the player gets 21, they win no matter what. Similarly, if the dealer “busts”, then the player automatically wins.

Our Blackjack Game
For our Blackjack game, the rules will be similar, but a few differences. Both the player and the dealer will begin with 0, and we will have a random number generator generate a number from 1-11 to simulate actual card values. If you (the player) think the number you got will not be high enough to beat the dealer, you will click “Hit Me” will will add another random number from 1-11 to the number you already have. You will continue doing this until you decide you have a number high enough that you think will beat the dealer or when you get 21+ in which you either win or bust. If you didn’t get 21 yet, but think you will bust on the next turn, you can choose to Reveal Cards to see if you won or not.

This homework will NOT give you the answers at the end as it is designed to test how much you have retained from previous labs. Therefore you may need to allow yourself time to go to homework help hours in case you get stuck!
A. HTML Layout

Step 1:
Create a Hw8 folder, and open NotePad++. Save the file as BlackJack.html.

Step 2:
1. Write out the HTML Skeleton (See Lab 1 if you have trouble with this).
   a. Add in the <style> and <script> tags within your head tag.

2. Inside the body you will need to use the <center>, <p>, <img>, <hr> and <table> tags. See the image below of how your page should look like.
   a. Use the attached Black Jack image found on the course website.
   b. The table will have attribute border="2" and contain 2 rows. In the first row one, you will use <th> tags for the column headers, and in the second row you will use <td> tags.
   c. For the first 3 <td> tags, put <p> tags in between the <td> tags and add &nbsp; in between the <p> tags. For the cell under the “Cash Remaining” column, put in value of $500, which is the amount of money we will start with.

When you finish adding all the HTML elements, you should get something like this:

Welcome to Black Jack!

<table>
<thead>
<tr>
<th>Your Total</th>
<th>Dealer Total</th>
<th>Bet Amount</th>
<th>Cash Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$500</td>
</tr>
</tbody>
</table>
B. CSS Styling
Things look a bit out of place right now, so we’ll need to add some CSS tags to make it look a little better.

Step 1:
Within your `<style>` tags, you will:
   1. Change the `td` to have a `width` of `120px`, a `height` of `50px`, a `background-color` of `orange`, and have all of the data inside `text-align` to the `center`.

   HINT ➜

   2. Set the `width` of the `hr` to be `50%`.
   3. Set the `width` and `height` of the `img` to be `320px` and `180px` respectively.

When you finish that, you should get something like this:

Welcome to Black Jack!

<table>
<thead>
<tr>
<th>Your Total</th>
<th>Dealer Total</th>
<th>Bet Amount</th>
<th>Cash Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$500</td>
</tr>
</tbody>
</table>
C. HTML Buttons
Now that we have our HTML layout and CSS styling in place, you’ll need to add buttons to give the user something to use to play the game.

Step 1:
For this page, we’re going to have SIX buttons – a button for each of the two bet amounts, a reset hand button, hit me button, a reveal cards button, and an add money button.

Explanation of Button Functionality
• $1: Adds $1 to bet.
• $5: Adds $5 to bet.
• Reset Hand: Resets hands and bets (does not reset cash remaining).
• Hit Me!: Adds random number from 1-11 until you “hit” 21 or you reveal your cards.
• Reveal Cards: Compares your/dealer totals and calculates winnings.
• Add Money: Prompts user for amount to add to their cash amount.

1. First, add a <p> tag where in it, you have “Game Result: ” followed by a <b> tag with the id of “resultId” after your table. This will act similarly to the ids for the decimal/binary/hex values from Lab 8. You will need this to display your game results later. It should look like:

   <p>Game Result: <b id="resultId"></b></p>

2. The buttons will be on two different rows. You will NOT be using any <table> or <tr> tags. Instead, you will separate each row using two <br/> tags. Add <button> tags with:
   o Row 1 contains the buttons: $1, $5, Reset Hand
   o Row 2 contains the buttons: Hit Me!, Reveal Cards, Add Money

3. Add an onclick=”” to each of the button tags.
4. Style the buttons in your CSS with width: 120px, height: 50px, and color: darkblue.

The result should now look like the following:
D. JavaScript Functionality
Like in the previous lab, you are going to use JavaScript to create the backbone of this game.

Step 1:
1. Inside your <head> tag, include a <script> tag much like we’ve done before.
2. For this page, we need a total of FIVE variables:
   - player – keeps track of player’s hand
   - dealer – keeps track of dealer’s hand
   - bet – current bet amount
   - cash – tracks remaining cash
   - name – your name
3. Set player, dealer, bet equal to 0, and cash equal to 500 (the starting amount). Set name equal to your name (in quotes).

Step 2:
In order to output values inside the each of the <td> cells, we need to use the <p> tags as a reference id. In our previous lab, we used <p id="decimalId">, "decimalId" being our id name. In this lab, we have 4 different ids, one for each of the cells (not including the “resultId” tag you added earlier).

Go back to your <td> tags and assign an id to each <p> tag inside the <td>, have the ids be:

“playerId” for Your Total
“dealerId” for Dealer Total
“betId” for Bet Amount
“cashId” for Cash Remaining

Your tags will look something like the following:

```html
<td><p id='playerId'></p></td>
```

Step 3:
Now that we have all of the structure in place let’s bring our buttons to life. When the $1 button is clicked, we want to add 1 to the value of bet, and also call the function showBet() to display our bet. To call a function, simply, type in the name of the function.

1. Add 1 to the value of bet with `bet++;`. Call the function showBet with `showBet();`.
2. Add 5 to the value of bet with `bet+=5;`. Call the function showBet with `showBet();`.

When you click on the $1 or $5 buttons, nothing should happen right now. Why? You CALL the functions when you click on the buttons, but you have not yet DEFINED the functions yet. Think of a function call as telling someone to build a window. If you haven’t given anyone instructions on how to build the window, as in, if you haven’t specified how the window should look like, what the dimensions are, or how the window should function, you can’t build the window properly.
We will define the contents of showBet() as follows:

```javascript
function showBet()
{
    document.getElementById("betId").innerHTML = "\$" + bet;
    if(cash - bet < 0)
    {
        alert("You can't bet that much! Add more money!");
    }
}
```

**What is this function doing?**

In the showBet() function, we look for the tag that contains the id of “betId” and puts the bet amount into the HTML at that tag. This effectively displays the bet. Then we check if the amount we bet is greater than the cash we have or not. If it is greater, we alert the user that they cannot bet that much money.

If you click on the $1 or $5, your bet amount should increment by how much money you added.

**Step 4:**

Now that the user can click buttons to decide how much to bet we need to give functionality to the Hit Me! button so we can actually play the game. We will be using JavaScript’s built in random number generator to simulate a supposed Black Jack game.

This is what clicking the Hit Me! button will do:

- When the player clicks the Hit Me! button, it will call the function `play()` so that the game will determine a random value between 1 (Ace) and 11 (face card) for the player, and add it to a previous value the player already may have gotten (starting from zero). The number will be shown on the screen.
- Then it will check if the player has already gotten 21 or more. If the player has, we will call the function `calculateWinnings()` to calculate the winnings or losses.
- Finally, it will determine the results as to who won, then display the proper result message as well as update the user’s cash if the user had won.

1. Create the following function:

```javascript
function play()
{
}
```

2. We first need to add a call to play(). When the Hit Me! button is clicked, we want to make a function call to `play()` with `play()`.
3. Also add a call to `calculateWinnings()`. When the Reveal Cards button is clicked, we want to make a function call to `calculateWinnings()`.
Step 5:
Now we will define the play() function.

1. Inside the function, use this line to generate a random number between 1 and 11, and add it to the current player value:
   
   ```javascript
   player += Math.floor((Math.random() * 11) + 1);
   ```

2. We need to show what number was generated for player. To do so, we will update the innerHTML of the playerId. How do we do this? Hint: Do you remember how the bet amount was displayed in showBet()? It is very similar.

3. Now check if the player value is \( \geq 21 \), and if it is, call the function calculateWinnings(), which we’ll define in a moment.

Step 6:
Let’s write the calculateWinnings() function. Note that all the code in Steps 6 and 7 will be written WITHIN the calculateWinnings() function.

To begin the calculateWinnings() function:

1. Create the function after the play() function (see image of the play() function skeleton from Step 4 if you don’t know how to create it).

2. We want to first calculate the card values that the dealer got. To do this, the dealer will continue adding a random number from 1-11 until the dealer reaches 18, in which the dealer will decide to stop.

Write a while loop in which the condition is \( \text{dealer} < 18 \). Inside the loop, write a line of code to generate a random number between 1 and 11, and add it to the current dealer value, seen below.

   ```javascript
   dealer += Math.floor((Math.random() * 11) + 1);
   ```

3. Deduct the bet amount from your total cash.
   
   Use the equation \( \text{cash} = \text{cash} – \text{bet} \).

4. We need to show what numbers were generated for dealer and player. To do so, we will update the innerHTML of the ids dealerId and playerId (same as last step).

Step 7:
Now that that’s done, we need to find out the winner of the game! There are FIVE cases to consider with priority:

1. If the player busts, the player loses no matter what (busted means more than 21). The result message you should display is: “Busted! You lose!”

2. If the dealer busts and the player hasn’t busted, then the dealer loses. The player will receive 2x the bet amount as winnings added to the cash total. The result message you should display is: “Dealer busted! ” + name + “ wins!”

3. If the player gets 21, then the player wins no matter what. They player will receive 3x the bet. The result message you should display is: name + “ got 21! ” + name + “ wins!”

4. If neither conditions 1-3 are true, then if the player has a larger value than the dealer, then the player wins. The player will receive 2x the bet. The result message you should display is: name + “ wins!”
5. If neither conditions 1-4 are true, then if the dealer has equal or greater value than the player, then the player loses. The result message you should display is: “Dealer wins!”

Note: the player’s bet will be deducted from the cash amount even if the player wins.

We will achieve this using a series of if/else if statements

If/else-if statements are a special case of if/else statements that making nesting them extremely easy. They show mutual exclusivity. A condition may only proceed if it has failed all previous tests, and will exit the test as soon as it becomes true.

The format of these statements is as show to the right:

```javascript
if (condition)
{
    stuff if true
}
else if (condition 2)
{
    stuff if true
}
else if (condition 3)
{
    stuff if true
}
```

And so on. So, to get you started, here are the first two statements and the skeleton for the other three.

```javascript
if(player > 21)
{
    document.getElementById("resultId").innerHTML = "Busted! You lose!";
}
else if(dealer > 21)
{
    document.getElementById("resultId").innerHTML = "Dealer Busted! " + name + " wins!";
    cash = cash + bet*2;
}
else if()
{
}
else if()
{
}
else if()
{
}
```

Can you figure out the rest?

Now there are only two more things you need to do in the calculateWinnings() function.

1. We want to display the new cash amount. Update the innerHTML of the id “cashId” to be equal to the cash variable.
2. We also want to reset our bet amount to 0 as in bet = 0.
Phew! Now all of that accomplishes what we need to for the calculateWinnings() function. At this point, test out if the Hit Me! and Reveal Cards functionality worked. You should see something similar to this when you click the Play button:

In this sample play, I clicked the Hit Me! button three times. The first time, I got an 11, the second time I got 4 added onto that, and the third time I got another 5 added onto that. At this point, I decided not to click Hit Me! anymore because the chances of getting a 1 to equal 21 were slim. Thus I pressed Reveal Cards to see what the dealer got. The dealer got 20 and I got 20, but since neither player got 21 or busted (over 21), and the dealer got an equal number than me, the dealer won.

There are 4 other cases. Make sure you are able to get the other cases when you click Hit Me!/Reveal Cards.

**Step 8:**
The next (and second to last) button we will implement is the Reset Hand button.

The resetHand() function will clear all the values (except cash), so that the player can get a fresh start.

1. **Create the function resetHand()** after the calculateWinnings() function.
2. We will clear the values displayed in each of the boxes. To do this, set the innerHTML of each of the id’s to the empty string ““.”
For example:
```
document.getElementById("playerId").innerHTML = "";
```

3. You will be doing this for “playerId”, “dealerId”, and “betId”.
4. For “cashId”, you will do the same thing, but instead of setting it equal to “”, you will set it equal to “$” + cash.
5. Set the player value, dealer value, and bet to be zero as in `player = dealer = bet = 0`.
6. Lastly, inside the onclick in the Reset Hand button, write a function call to `resetHand()`.

Try this out by placing a bet and clicking Hit Me!. Then click the Reset Hand button. The boxes for your total, the dealer total, the bet amount, and the result should be blank. Cash should not change.

**Step 9:**
For the last thing, we need to just make our “Add Money” button function correctly. We will be implementing one last function – the `addMoney()` function.

1. In the onclick for the Add Money button, add a function call to `addMoney()`.
2. Now we need to define the addMoney() function. Create the function `addMoney()` like how we created the other functions.
3. Inside the function, create a new variable called `value` and set it equal to a `prompt` asking users “How much money would you like to add?”.
4. Add `value` to `cash` using the following line of code. The function `parseInt` converts the string value from the user input into an integer.
```
cash += parseInt(value);
```
5. Finally, display the cash total by setting the innerHTML of “cashId” to “$” + cash.

**Step 10:**
That’s it! You’re all done! Now all you have to do is test it and make sure everything works. You should be able to place bets of varying dollar amounts, and it should be subtracted from your cash remaining properly. You should win some games versus the dealer, and lose some. Your cash reserves should be updated after each game accordingly.

**E. (Optional) Error Checking**
You might have noticed that you can reach situations that you shouldn’t be able get to. For example, when the player has no money left, we throw an alert that says you can’t bet that much because that’s more than the amount of money you have left. However, we don’t prevent you from betting or playing the game. You shouldn’t be able to play the game if you have no money.

There are other cases you can consider. Some examples of problems include:
1. You can add negative amounts of money or money that is not an integer.
2. You can keep clicking “Hit Me!” after the game is over.
3. You can keep clicking “Reveal Cards” after the game is over.

If you are interested in doing any kind of programming in the future, preventing errors is an integral part of the process. We don’t do this in the homework to simplify it for you, but if you are up for a challenge, then this is for you.
F. Putting it all online

Step 1:
Modify your **CSE3Page.html** to include a link to your **BlackJack.html**.

Step 2:
Put everything online and get checked off.

REMEMBER to either drag your ENTIRE CSE3 folder into your **public_html** folder OR into the WHITESPACE inside the **public_html** folder!!!!

Checkoff:

- Ensure **BlackJack.html** is linked on your CSE 3 Page.
- Demonstrate that all your buttons work properly.
- Demonstrate that you can achieve the **FIVE** conditions that you can reach by clicking the Hit Me!/Reveal Cards buttons.
- Ensure that each of your five conditions display separate result messages.