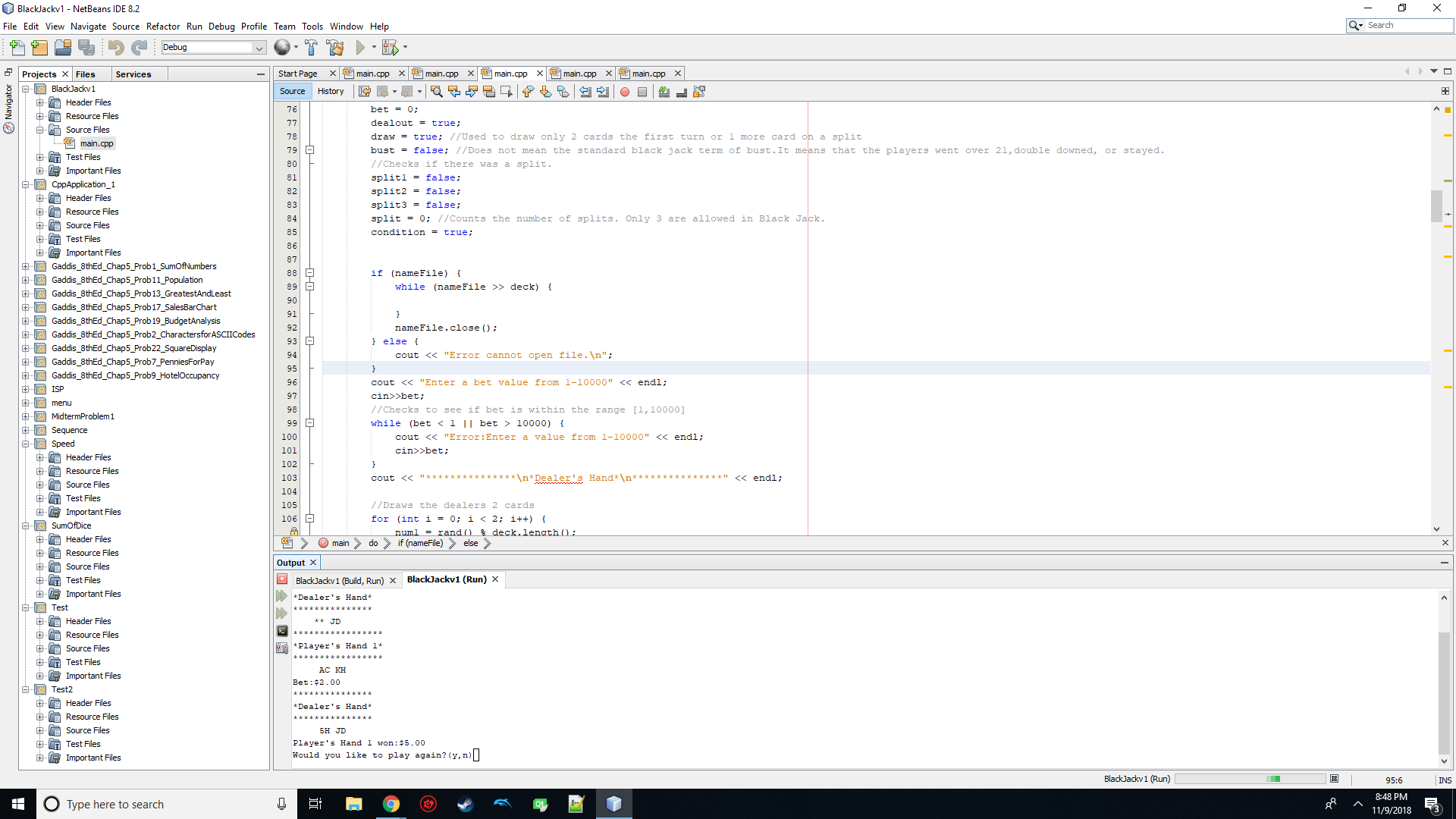
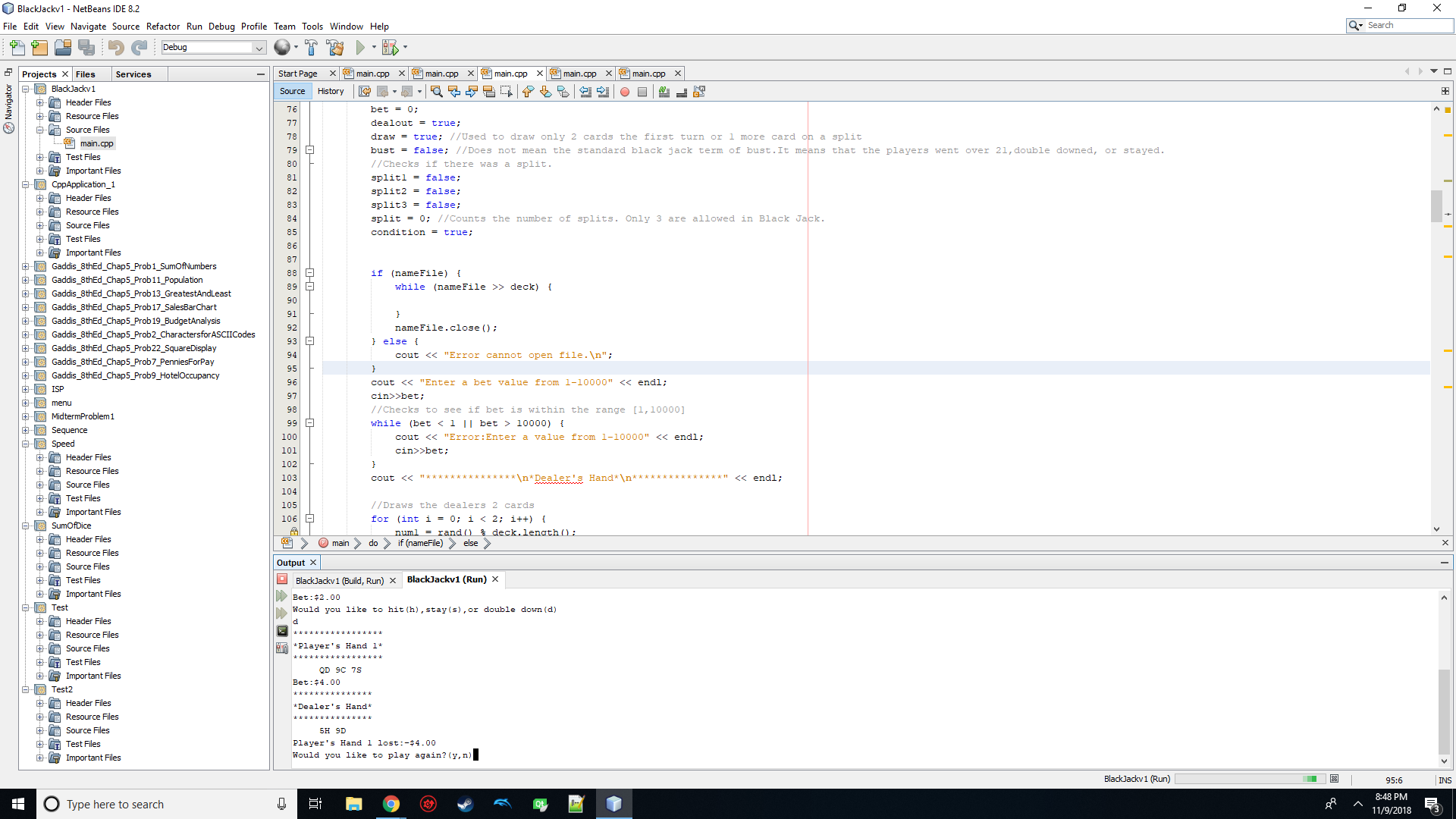
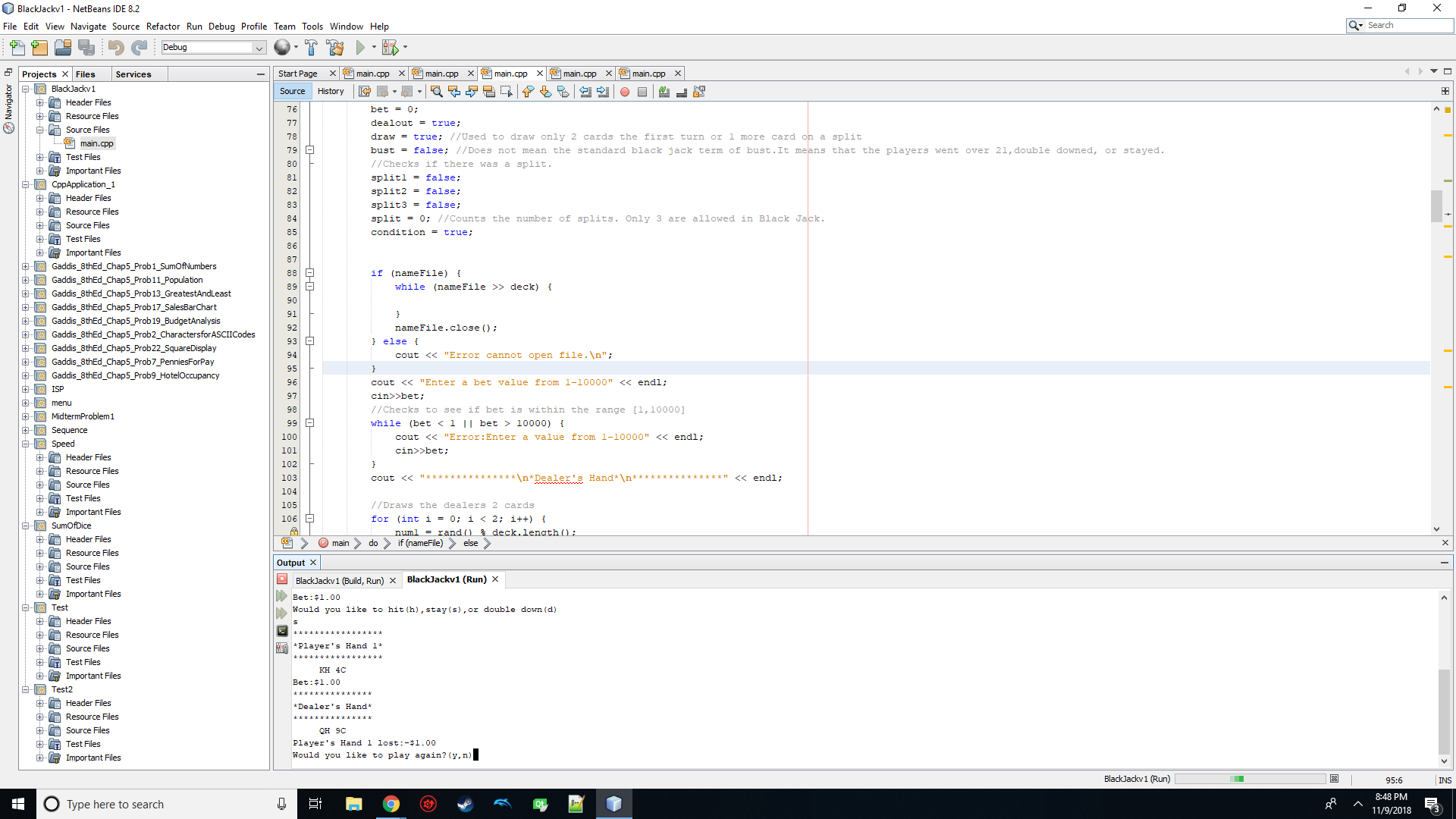
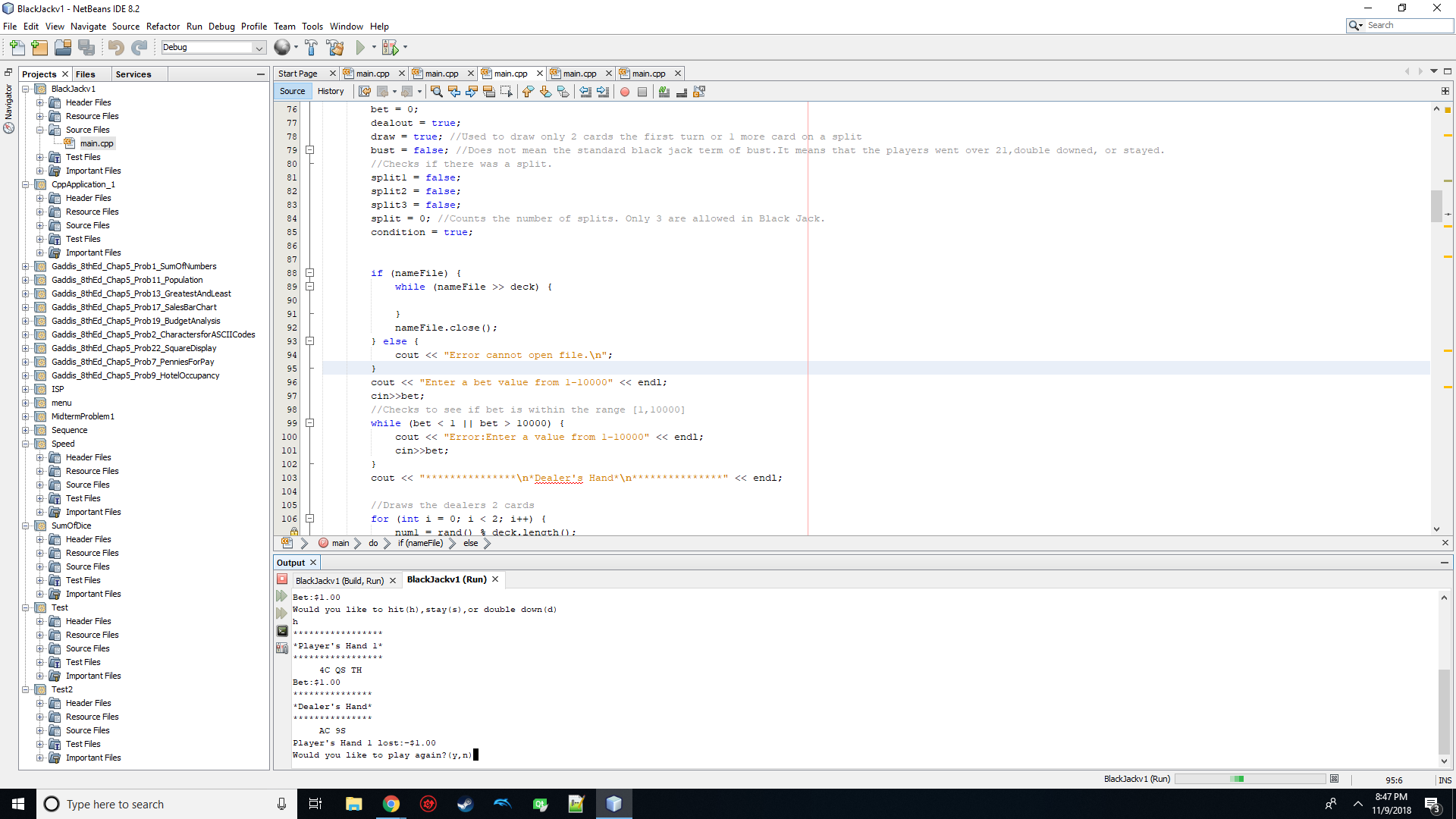
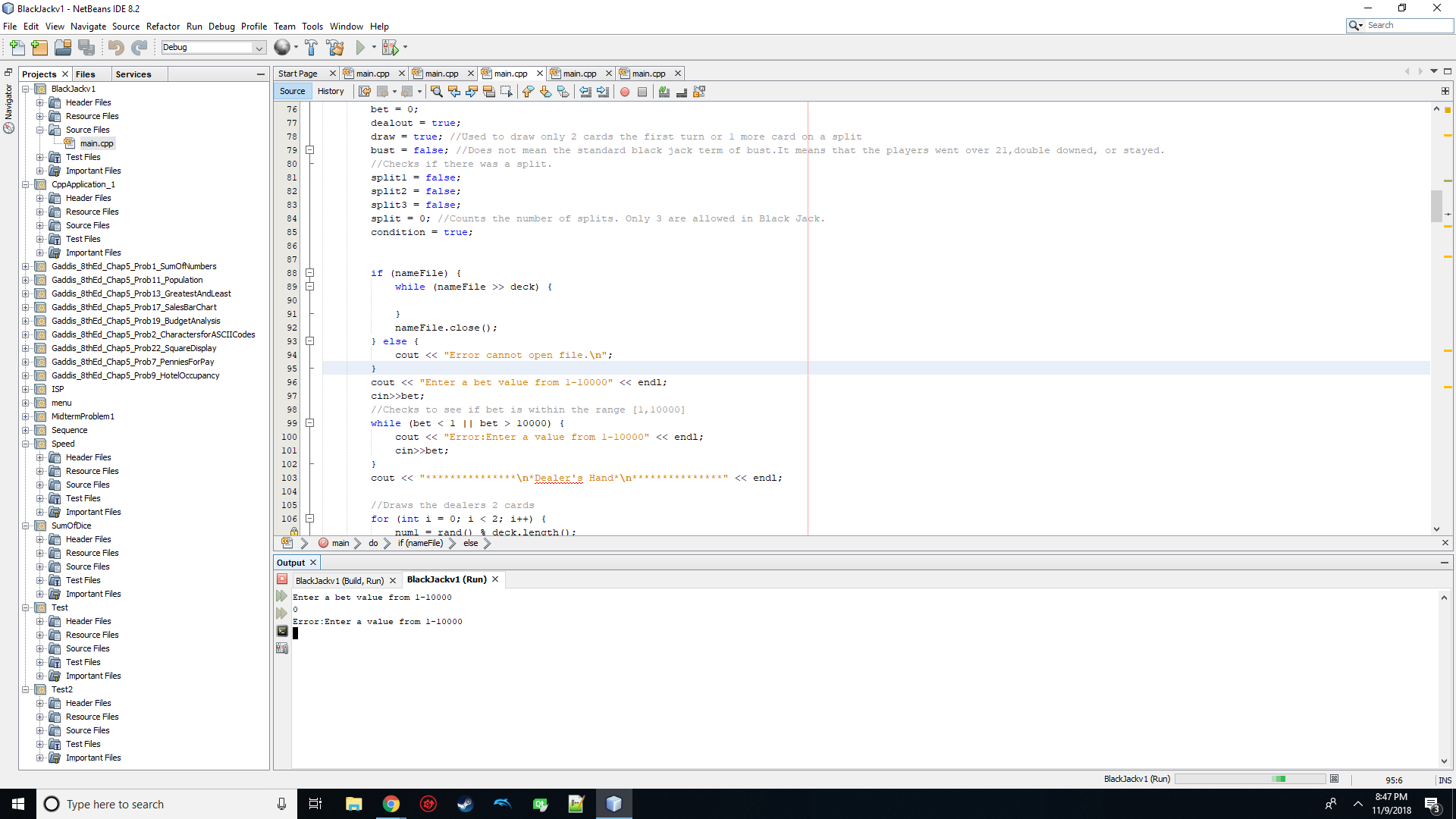
Eric Perez

11/9/18

Project 1 Black Jack v1

Rules:

1. The player must get less than or equal to 21 and greater than the dealer in order to win.
2. If a player gets more than 21 they lose.
3. The player is only able to split 3 times.
4. If a player splits they put an equal bet value on the split.
5. If a player hits they are given 1 more card. They can keep hitting until they bust.
6. If a player doubles down they double their bet and are given only one more card.
7. If a player doubles down and splits they give are given 2 more cards instead of one.
8. If a player stays they are given no more cards.



while (bet < 1 || bet > 10000) {

cout << "Error:Enter a value from 1-10000" << endl;

cin>>bet;

}

I used the above code to check if the user entered a bet value between 1-10000.The while loop will keep executing until the condition is false.Unfortunately, it does not check if the user entered anything other than an integer. If for example the user entered a character the program will crash.

if (draw) {

if (split1 || split2 || split3) {

j = 1;

}

for (int i = 0; i < j; i++) {

num1 = rand() % deck.length();

if (num1 % 2 != 0) {

num1 = num1 - 1;

}

card = deck.substr(num1, 1);

suit = deck.substr(num1 + 1, 1);

deck.erase(num1, 2);

player = player + (card + suit);

}

//Used to format the cards with a space in between.

cout << hand << endl;

cout << setw(7);

for (int i = 0; i < (player.length()) / 2; i++) {

cout << player.substr(num2, 2) << " ";

num2 = num2 + 2;

}

cout << endl;

cout << "Bet:$" << bet << endl;

draw = false;

}

The first if statement is used to draw the first two cards in the game. The second if statement changes the j value to 1 so the for loop will execute only once instead of twice. This only occurs when the boolean split1,split2,or split3 are true. The line num1 = rand() % deck.length(); will be used many times in the program. It is used to generate a random number from 0-104.

Every card value will land on an even number and every suit will fall on an odd number.

Eg. 2H2S2D2C3H3S3D3C

To prevent the program from selecting a suit instead of a card value I used the if statement (num1%2!=0) followed by decrementing num1 by 1 so any odd number will be converted into an even number.

card = deck.substr(num1, 1);

suit = deck.substr(num1 + 1, 1);

deck.erase(num1, 2);

The string card will be initialized to a substr of deck that is the card’s value. It will then erase that card from the deck. For example if the partial deck above was used and the card value was 2 with the suit H then the partial deck would be 2S2D2C3H3S3D3C.

for (int i = 0; i < (player.length()) / 2; i++) {

num3 = 0;

count = 0;

check1 = player.substr(num2, 1);

for (int j = 0; j < (player.length()) / 2; j++) {

check2 = player.substr(num3, 1);

if (!check1.compare(check2)) {

count = count + 1;

if (count > 1 && split < 3 && condition) {

cout << "Would you like to split?(y/n)" << endl;

cin>>answer;

if (answer == 'y') {

split = split + 1;

table = player.substr(num3, 2);

card = player.substr(num3, 1);

player.erase(num3, 2);

if (!card.compare("T") || !card.compare("J") || !card.compare("Q") || !card.compare("K")) {

total2 = total2 - 10;

} else if (!card.compare("A")&&((total2 + 11) <= 21)) {

total2 = total2 - 11;

} else if (!card.compare("A")&&((total2 + 11) > 21)) {

total2 = total2 - 1;

} else {

total2 = total2 - stoi(card);

}

cout << hand << endl;

cout << setw(10) << player << endl;

if (split == 1) {

split1 = true;

} else if (split == 2) {

split2 = true;

} else if (split == 3) {

split3 = true;

}

condition = false;

} else {

condition = false;

}

}

}

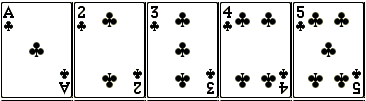
num3 = num3 + 2;

}

num2 = num2 + 2;

}

The above code took the most time to figure out without using arrays. It checks if a split is possible.



In this case it is since we have two aces. The first for loop iterates only after the second for loop is done. We fix the first card value while the second for loop iterates through all of the cards comparing if the two cards are of the same value. If it finds that two cards are the same then the program will then ask the user if they would like to split. The problem with this sorter is that it will compare a card to every card in the player’s hand including comparing the card to itself. Therefore, no matter what the player’s hand is there will always be a card that is the same. A work around for this is to include a variable count. We know that count will always be 1 or more.It will be 1 if there are no two of the same cards not including itself. We set a condition for count to be greater than 1 for the program to proceed to ask if the user wants to split the cards. Another condition we set is that split has to be less than 3 since it is a rule in Black Jack that the payer cannot split more than 3 times in a game. The final condition is used to prevent the program asking if the user wants to split twice.