Project 1

Max Lothringen
12/10/2018
CSC-5-47993

BlackJack Program

Rules of BlackJack

1. Blackjack may be played with one to eight decks of 52-card decks.
2. Aces may be counted as 1 or 11 points, 2 to 9 according to pip value, and tens and face cards count as ten points.
3. The value of a hand is the sum of the point values of the individual cards. Except, a "blackjack" is the highest hand, consisting of an ace and any 10-point card, and it outranks all other 21-point hands.
4. After the players have bet, the dealer will give two cards to each player and two cards to himself. One of the dealer cards is dealt face up. The facedown card is called the "hole card."
5. If the dealer has an ace showing, he will offer a side bet called "insurance." This side wager pays 2 to 1 if the dealer's hole card is any 10-point card. Insurance wagers are optional and may not exceed half the original wager.
6. If the dealer has a ten or an ace showing (after offering insurance with an ace showing), then he will peek at his facedown card to see if he has a blackjack. If he does, then he will turn it over immediately.
7. If the dealer does have a blackjack, then all wagers (except insurance) will lose, unless the player also has a blackjack, which will result in a push. The dealer will resolve insurance wagers at this time.
8. Play begins with the player to the dealer's left. The following are the choices available to the player:
    - Stand: Player stands pat with his cards.
    - Hit: Player draws another card (and more if he wishes). If this card causes the player's total points to exceed 21 (known as "breaking" or "busting") then he loses
9. After each player has had his turn, the dealer will turn over his hole card. If the dealer has 16 or less, then he will draw another card. A special situation is when the dealer has an ace and any number of cards totaling six points (known as a "soft 17"). At some tables, the dealer will also hit a soft 17.
10. If the dealer goes over 21 points, then any player who didn't already bust will win.
11. If the dealer does not bust, then the higher point total between the player and dealer will win.
12. Winning wagers pay even money, except a winning player blackjack usually pays 3 to 2. Some casinos have been short-paying blackjacks, which is a rule strongly in the casino's favor.

My Program

I chose to play with one deck against a AI. This AI or dealer was a completely random one that had no real strategy. This game involves mostly luck and very little skill involved. Initially the user is given 100$ to play and is allowed to bet however much they wish. If the user reaches 0$ then they lose and can no longer play; however if they reach 1000$+ they win and can continue playing indefinitely. Their first two cards will be given to the user along with the card point total. The user can then choose to draw another card until they reach a total count of over 21 or stick with their current deck. From there the dealers cards will be revealed along with the dealers choices. Then whoever is over 21 loses or whoever is closer to 21. Else the game is a draw. If the user wins their bet is added to their winnings; however if the user loses then the bet is subtracted from their winnings. This process continues until the user no longer wants to play or goes broke.

Basic Info

Lines of code: about 260
Number of variables: about 20
I am not that close to finishing my program as I would like to add ai difficulty and a better format for the initial menu. Also the game is never ending if the user reaches either broke or wins over 1000$, which is something I would like to fix.

Pseudo-Code

//IN - card name
//IN - drawer's total points

//Execution begins with main
//title
//spacing

//players cash amount
//player card count
//dealer card count
//if the user chooses to not hit
//if the player does not want to continue
//the card name in reference to the array
//dealer card name in reference to the array

//the worth of the bet that the user chose

// INPUT -- get the desired amount the user wants to bet.

//PROCESSING -- Subtracts the user's desired bet amount from their current balance

//PROCESSING -- calls the function draw_card to determine the player's count.

//OUTPUT -- Displays the player's cards

//INPUT -- Asks the player if they want another card

//PROCESSING -- checks if the user chooses to get another card and draws a card.

//PROCESSING -- adds the new cards count to the current player's count

//OUTPUT -- Displays the player's cards

//PROCESSING -- Sees if the user's total count goes over 21.

//OUTPUT -- Displays the player's count

//PROCESSING -- Draws two cards for the dealer.

//PROCESSING -- Adds the cards count to the dealer's current count.

//OUTPUT -- Displays the dealer's current cards.

//OUTPUT -- Displays the dealer's current count.

//PROCESSING -- Checks if the dealers count is less than 16 and will get new cards. Also checks if the player has 21 already

//PROCESSING -- adds the new cards count to the dealers's current count

//OUTPUT -- Displays the dealer's count.

//PROCESSING -- checks for win, lose, and tie conditions

//OUTPUT -- Displays a message saying that there is a tie.

//PROCESSING -- Adds the user's initial bet back

//OUTPUT -- Displays a message saying that the player wins

//PROCESSING -- Returns the initial bet * 2

//OUTPUT -- Displays a message saying the the player loses.

//INPUT -- Asks if the user would like to play again.

//PROCESSING -- checks if the user says no and if so ends the game.

//PROCESSING -- checks the user has over 1000 or 0

//OUTPUT -- displays a winning message.

//OUTPUT -- displays a losing message.

//PROCESSING -- checks if the users reaches below 0, above 1000, or chooses to leave the game.

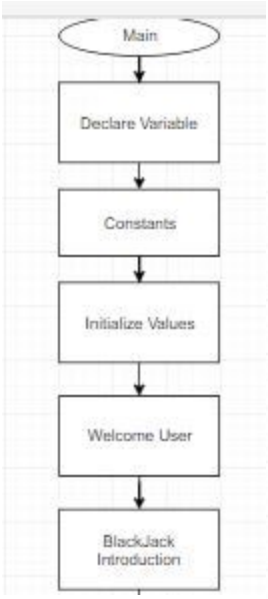
//card info
//current drawer's points

//Generates a random number to choose the card value and the suit.

//Creates a string to display the card name

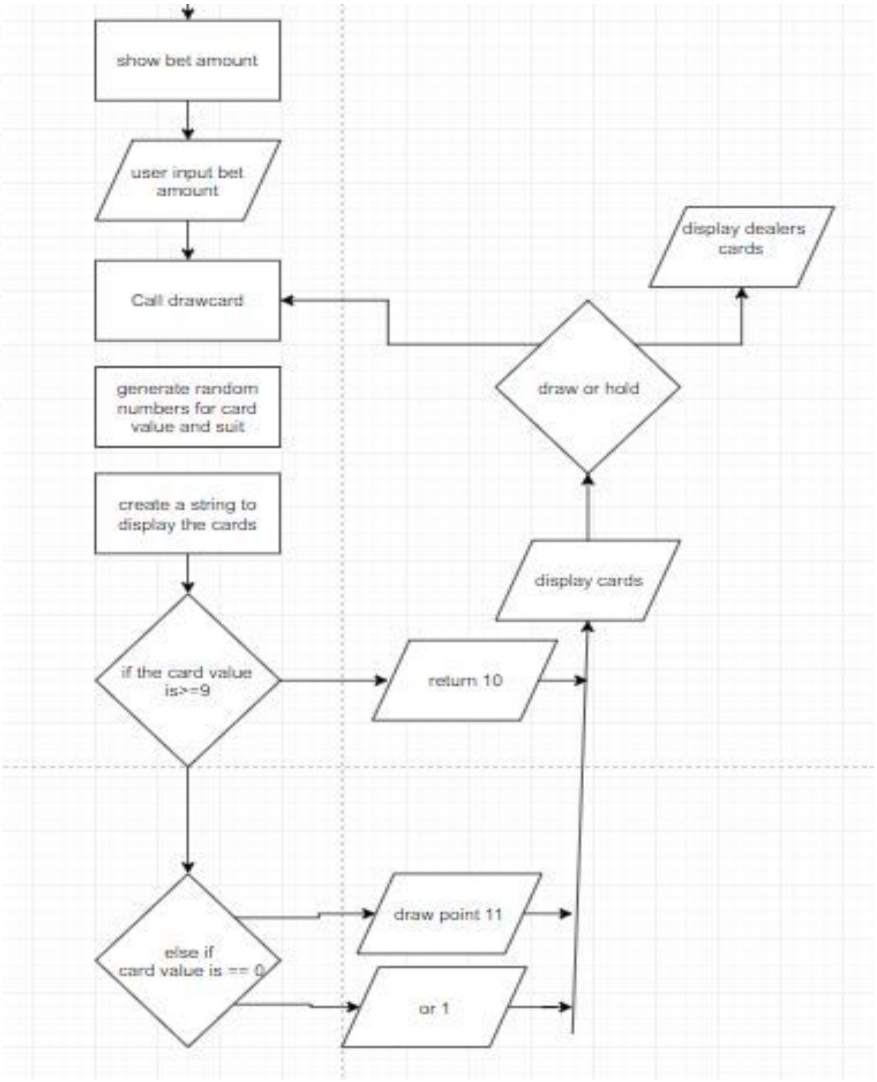
//PROCESSING -- checks the card value and returns the value.

Flowchart

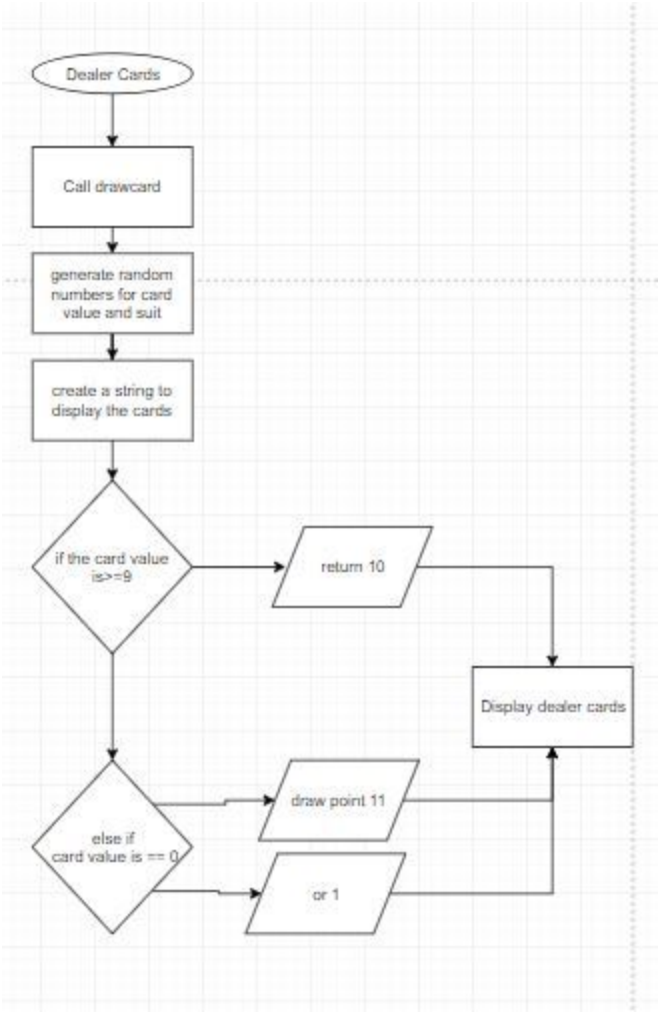


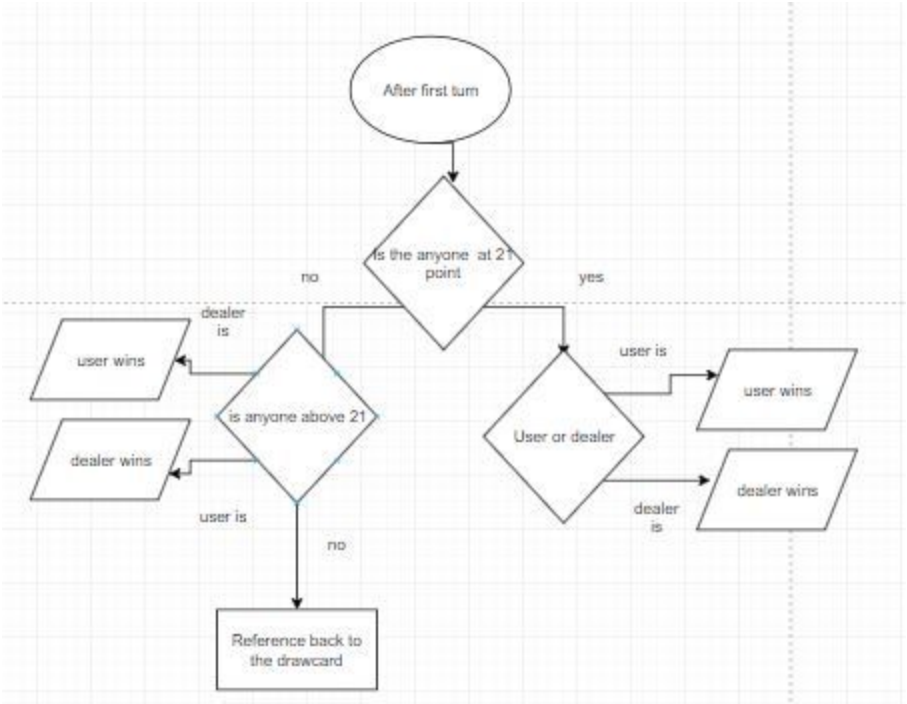
This flowchart is the start of the program and basically introduces it and states all the basics.

This flowchart ------> Starts by showing the basic pathway of the game.

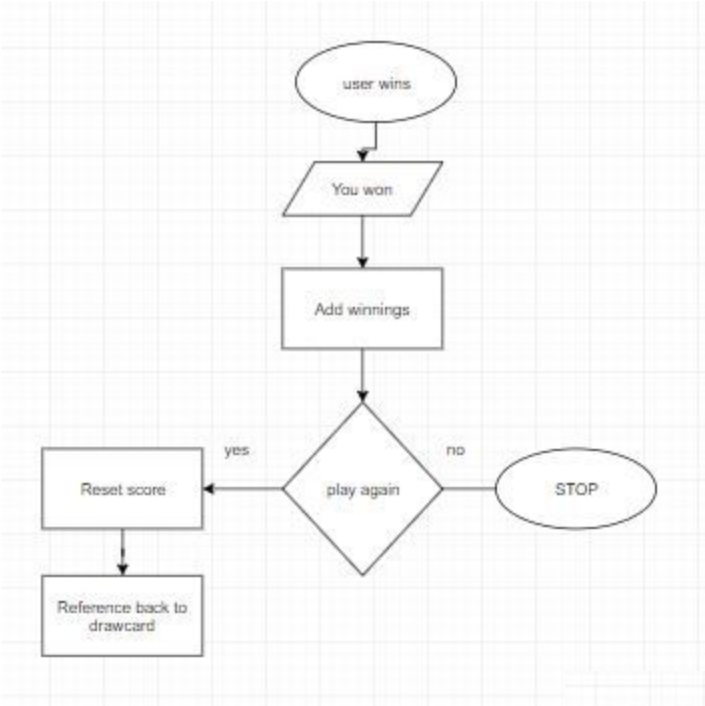


This flowchart shows the process for getting the dealers cards to display



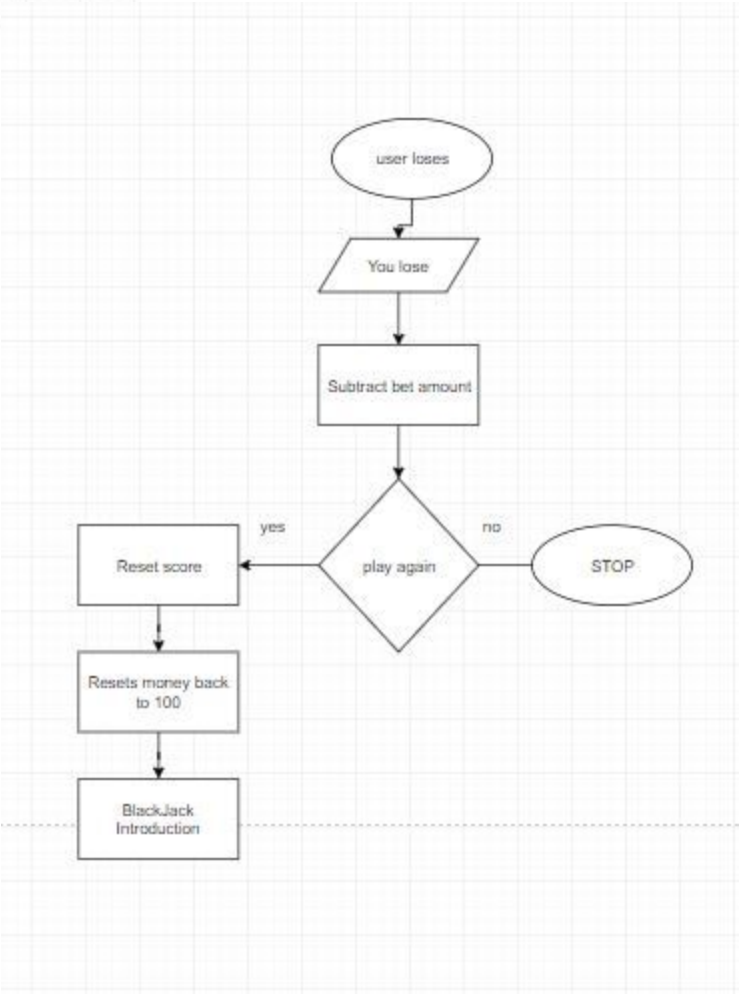


This flowchart checks the points and winners after the first drawing. This will repeat until either the dealer or user gets to or above 21 points.



If the user wins the game then they will be asked to play again. If they want to they will continue playing with the winnings being added to their bet amount.

If the user loses than they lose their money. But can play again with a full reset of score and money back to the base amount.



Variables

```
int plyCash;        //players cash amount
int plyCnt;         //player card count
int dealCnt;        //dealer card count
bool noHit;         //if the user chooses to not hit
bool plyGone;       //if the player does not want to continue
string plyCard;     //the card name in reference to the array
string delCard;     //dealer card name in reference to the array
int betWrth;        //the worth of the bet that the user chose
char temp           //temporary choice
string cardNam      //card name
char resp;          //response
int cardVal         // card value
int cardSui         //card suit
string CARDV        //card value in array
string CARDS        //card suit in array
```

Program

```
/*
 * File:   main.cpp
 * Author: Max Lothringen
 * Created on November,3 2018, 10:07 PM
 * Purpose Blackjack program
 */

// System Libraries
#include <iostream>
#include <vector>
#include <cmath>
#include <stdlib.h>
#include <ctime>
using namespace std;
// User Libraries

// Global Constants  Physics/Math/Conversions/Array Dimension

// Function Prototypes
void BubbleSort(int *a, int size);
int draw_card(string &card,  // IN - card name
              int drawpts);  // IN - drawer's total points
void printNumber(int u);
void printNumber(float u);
// Execution begins with main
int main() {
  srand(time(NULL));

  cout << "BlackJack" << endl;  // title
  cout << "*********" << endl;  // spacing
```

```
cout << endl;
cout << "Welcome to BlackJack" << endl;
vector<int> g1;
int plyCash;  // players cash amount
int plyCnt;  // player card count
int dealCnt;  // dealer card count

bool noHit;  // if the user chooses to not hit
bool plyGone;  // if the player does not want to continue

string plyCard;  // the card name in reference to the array
string delCard;  // dealer card name in reference to the array

plyGone = false;
plyCash = 100;
int sCash = plyCash;
do {
  int betWrth;  // the worth of the bet that the user chose

  plyCnt = 0;
  dealCnt = 0;

  plyCard = "";
  delCard = "";

  noHit = false;

  do {
    //  INPUT -- get the desired amount the user wants to bet.
    cout << "You have $" << plyCash << ". Enter bet:" << endl;
    cin >> betWrth;
  } while (betWrth < 1 || betWrth > plyCash);

  // PROCESSING -- Subtracts the user's desired bet amount from their current
  // balance
  plyCash -= betWrth;

  // PROCESSING -- calls the function draw_card to determine the player's
  // count.
  for (int i = 0; i < 2; i++) {
    string cardNam = "";
```

```
  plyCnt += draw_card(cardNam, plyCnt);
  plyCard += cardNam;
  int tst = plyCnt;
  float test = plyCash;
  printNumber(tst);
  printNumber(test);
}

// OUTPUT -- Displays the player's cards
cout << "Your Cards are:\n";
cout << plyCard;

if (plyCnt < 21) {
  do {
    // INPUT -- Asks the player if they want another card
    char temp;
    cout << "Your total is " << plyCnt
         << ". Do you want another card (y/n)?\n";
    cin >> temp;

    // PROCESSING -- checks if the user chooses to get another card and
    // draws a card.
    if (temp == 'y') {
      string cardNam;

      // PROCESSING -- adds the new cards count to the current player's
      // count
      plyCnt += draw_card(cardNam, plyCnt);
      plyCard += cardNam;

      // OUTPUT -- Displays the player's cards
      cout << "Your Cards are:\n";
      cout << plyCard;

      // PROCESSING -- Sees if the user's total count goes over 21.
      if (plyCnt > 21) {
        // OUTPUT -- Displays the player's count
        cout << "Your total is " << plyCnt << endl;
        cout << "Your count is over 21. That's a bust!\n";
        noHit = true;
      }
```

```
    } else {
      noHit = true;
    }
  } while (!noHit);
}

// PROCESSING -- Draws two cards for the dealer.
for (int i = 0; i < 2; i++) {
  string cardNam = "";

  // PROCESSING -- Adds the cards count to the dealer's current count.
  dealCnt += draw_card(cardNam, dealCnt);
  delCard += cardNam;
}

// OUTPUT -- Displays the dealer's current cards.
cout << "\nThe Dealer's cards are:\n";
cout << delCard;

// OUTPUT -- Displays the dealer's current count.
cout << "The Dealer's count is: " << dealCnt << endl;

// PROCESSING -- Checks if the dealers count is less then 16 and will get
// new cards. Also checks if the player has 21 already
while (dealCnt <= 16 && plyCnt <= 21) {
  string cardNam = "";

  // PROCESSING -- adds the new cards count to the dealers's current count
  dealCnt += draw_card(cardNam, dealCnt);

  // OUTPUT -- Displays the dealer's count.
  cout << "The Dealer draws a card!\n" << cardNam;
  cout << "The Dealer's count is: " << dealCnt << endl;
}

// PROCESSING -- checks for win, lose, and tie conditions
if (dealCnt == plyCnt) {
  // OUTPUT -- Displays a message saying that there is a tie.
  cout << "Both of you have the same count. Tie!\n\n";

  // PROCESSING -- Adds the user's initial bet back
```

```
  plyCash += betWrth;

} else if (plyCnt > dealCnt && plyCnt <= 21 ||
          plyCnt <= 21 && dealCnt > 21) {
  // OUTPUT -- Displays a message saying that the player wins
  cout << "You win!\n\n";

  // PROCESSING -- Returns the initial bet * 2
  plyCash += betWrth * 2;

} else {
  // OUTPUT -- Displays a message saying the the player loses.
  cout << "\nYou lose!\n\n";
}

char resp;

// INPUT -- Asks if the user would like to play again.
cout << "Play again?(y/n):\n";
cin >> resp;

// PROCESSING -- checks if the user says no and if so ends the game.
if (resp == 'y') {
  plyGone = true;
} else
  exit(0);

// PROCESSING -- checks the user has over 1000 or 0
if (plyCash >= 1000) {
  // OUTPUT -- displays a winning message.
  cout << "YOU HAVE OVER $1000. YOU WIN!!!\n\n\n";

} else if (plyCash == 0) {
  // OUTPUT -- displays a losing message.
  cout << "You have $0. GAME OVER\n\n\n";
  plyGone = false;
  cout << "You have no more money. Sorry." << endl;
  return false;
}

// PROCESSING -- checks if the users reaches below 0, above 1000, or chooses
```

```
    // to leave the game.
  } while (plyCash > 0 || plyCash <= 1000 || plyGone);

  return 0;
}

int draw_card(string &card,  // card info
              int drawpts)  // current drawer's points
{
  string CARDV[13] = {"Ace",   "Two",  "Three", "Four", "Five",  "Six", "Seven",
                      "Eight", "Nine", "Ten",   "Jack", "Queen", "King"};
  string CARDS[4] = {"Diamonds", "Clubs", "Heart", "Spades"};
  // Generates a random number to choose the card value and the suit.
  int cardVal = rand() % 13;
  int cardSui = rand() % 3;
  static int shVal;
  vector<int> g1;
  // Creates a string to display the card name
  card = "    " + CARDV[cardVal] + " of " + CARDS[cardSui] + "\n";

  // PROCESSING -- checks the card value and returns the value.
  if (cardVal >= 9) {
    return 10;
  } else if (cardVal == 0) {
    if (drawpts > 11) {
      return 1;
    } else {
      return 11;
    }
  }
  return cardVal + 1;
}
void BubbleSort(int *a, int siz)  // template for the card sorting
{  // works but not with card need to add
  int i, j, k, temp;
  for (i = 0; i < siz - 1; i++) {
    for (j = 0; j < siz - 1; j++) {
      if (*(a + j) > *(a + j + 1)) {
        temp = *(a + j + 1);
        *(a + j + 1) = *(a + j);
        *(a + j) = temp;
```

```
      }
    }
    for (k = 0; k < siz; k++) cout << *(a + k) << " ";
    cout << endl;
  }
  int dum = 1;
}
void printNumber(int u) {  // playcount overloading
  cout << endl;
};
void printNumber(float u) {  // playercash overloading
  cout << "";
};
```