```
//============================================================================
// Name        : Blackjack.cpp
// Author      : douglas holmes
// Version     : 1.0
// Copyright   : ©holmes2018
// Description : Blackjack
//============================================================================
/**************************************************************************************
Blackjack C++ routine created by Douglas Holmes 11-2018 to 12-2018 ©holmes2018.
This game is based on the instructions for blackjack on the “Bicycle cards” web site.
One player, plays the computer dealer, and can split up to four times.  Chips are
offered at the beginning of the game and can be added during play.  Hit, stay, split,
double down, and print-hand are commands the player can choose. Six new ordered decks
of cards are opened and shuffled four times.  The player is then asked to cut them
before they are placed in a shoe (stack).  This program was created using the Eclipse
compiler on a Mac.  I created this program as a final project for a MIT Open Courseware
class #6-096.
***************************************************************************************/


#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
//#include <cwchar>
//#include <string>
//#include <wchar.h>
using namespace std;
// global variable
int totalplayers = 1; // player
int splittime[6] = {0}; // [0] is used to set the hand that is going to be split
int cardT;
int bump;

//declare prototypes
// a struct to create "shoe" of cards
typedef struct { int *stk ; int maxelem; int top; } STACK;
STACK *init(int n);
void fillStack(STACK *s, int arr[], int x);
void push(STACK *s, int data);
int pop(STACK *s);
void deleteNode(STACK *s, int data);
void hit(STACK *s, int pno, int a[], int count[pno]);
void dealCards(STACK *s, int player[], int house[], int count[]);
void CheckSplit (STACK *s, int a[], int b[], int count[totalplayers], double bet[], int splitplayer);
string cardName(int a[]);
string cardNamed(int a);
int cardValue(int a[]);
int winner(int a[], int bank, double bet[], int natural[]);
void newDeck (int x, int arr[x]);
void printArray (int cardArr[], string whoIs, int pno, double bet[pno], int n);
void swap (int *a, int *b);
void randomize ( int arr[], int n);
void cutCards(int arr[], int x, int c);
int checkNatural(int a[], int b[], double bet[], int pno);
int cardDealer(int a);
void doubleDown(STACK *s, int pno, int a[], int count[], double bet[]);


// Driver program to test above function.
int main()
{
    string hand[6]={"Dealer   ", "Original ", "Split 1 ", "Split 2 ", "Split 3 ", "Split 4 "};
    int x = 52 * 6;
    cardT = x;
    int w = 0;
    int c;
    double bet[6]={0};
    int bank = 20;
    int reshuffling;
    char resp;
    int *p;
    int *p1;
    int *cc;
    string CreditCard;
    STACK *s = NULL;
    int arr[x];
    s=init(x);
    int n = sizeof(arr)/ sizeof(arr[0]);

    cout <<"Blackjack based on Bicycle playing car rules (6 decks of cards)........\n\n";
    newDeck (x, arr);
    randomize (arr, n);
    randomize (arr, n);
    randomize (arr, n);
    randomize (arr, n);
    cout << "Cut cards, pick a number between 1 and " << x << ": \n";
    while(!(cin >> c))
        {
            cout<<"Invalid data type! Please enter 'a number' between 1 and " << x;
            cin.clear();
            cin.ignore(999,'\n');
        }
    if(c < 1 || c > x){c = 1;}

    cutCards(arr, x, c);
    fillStack(s, arr, x);
    srand (time(NULL)); /* initialize random seed: */
    reshuffling = rand() % 25 + 26;/* generate a number between 25 and 50: */
    cout << "The Card Shoe is filled and the reshuffling card has been placed.\n\n";
    cout << "Chips have the value of $10.00, How many chips would you like to Buy? \n";
    while(!(cin >> bank))
         {
             cout<<"Invalid data type! Please enter 'a number'";
             cin.clear();
             cin.ignore(999,'\n');
         }

    cout << "Please put in your Credit Card Number:";
    while(!(cin >> CreditCard))
         {
             cout<<"Invalid data type! Please enter 'a number'";
             cin.clear();
             cin.ignore(999,'\n');
         }


    cout << "processing order \n";
    int j = 0;
    while(j < 100){cout << "."; usleep(10000); j++;}
    cout << "\n";
    cout << "That is not a valid credit card, leave your watch and I'll let you play anyway........\n\n";
    cout << "You have " << bank << " chips, Please place your bet:";
    while(!(cin >> bet[1]))
             {
                 cout<<"Invalid data type! Please enter 'an even number' between 1 and " << bank;
                 cin.clear();
                 cin.ignore(999,'\n');
             }

while(bet[1] != 0)
{
    int insurance[3] ={0};
    int natural[6] = {0};
    int dealerhit = 0;
    int count[6]= {0};
    int scores[6]={0};
    int house[10] = {0};
    int player[10] = {0};
    int split1[10] = {0};
    int split2[10] = {0};
    int split3[10] = {0};
    int split4[10] = {0};
    totalplayers = 1;
    splittime[0] = 1;
    splittime[1] = 1;
```

```
int currentplayer = 1;
w = 1;
p = player;
p1 = house;
cc = count;
dealCards(s, p, p1, count);
printArray (p1, hand[0] ,0, bet, 0);
// force Split/natural
//   p[0] = 1;
//   p[1] = 14;
printArray (p, hand[currentplayer],currentplayer , bet,  w);
insurance[0] = cardDealer(house[0]);
insurance[1] = cardDealer(house[1]);
if(insurance[0] == 11)
{
cout << "Would you like to buy insurance (input a value between 0 and " << bank - bet[1] << " )?\n";
while(!(cin >> insurance[2]))
     {
         cout<<"Invalid data type! Please enter 'a number'";
         cin.clear();
         cin.ignore(999,'\n');
     }
bank -= insurance[2];
if(insurance[0] + insurance[1] == 21 && insurance[2] > 0 )
{
    bank += insurance[2] * 2;
    cout << "Good thing you had insurance, the dealer has 21, your bank is now " << bank << " chips\n";
    currentplayer = totalplayers + 1;
}else if(insurance[0] + insurance[1] == 21 && insurance[2] == 0 )
{
        bank -= insurance[2] + bet[1];
        cout << "Should have had some insurance dealer has 21, your bank is now " << bank << " chips\n";
        currentplayer = totalplayers + 1;
}else if(insurance[0] + insurance[1] != 21 && insurance[2] > 0 )
    {
            bank -= insurance[2];
            cout << "Dealer does not have 21 you loose your side bet, your bank is now " << bank << " chips\n";
}else
{cout << "dealer does not have 21 \n";}
}
splittime[0] = 1;
while(currentplayer <= totalplayers)
{
if(currentplayer == 1){p = player;}
if(currentplayer == 2){p = split1;}
if(currentplayer == 3){p = split2;}
if(currentplayer == 4){p = split3;}
```

```
    if(currentplayer == 5){p = split4;}
    if(totalplayers == 1){p1 = split1;}
    if(totalplayers == 2){p1 = split2;}
    if(totalplayers == 3){p1 = split3;}
    if(totalplayers == 4){p1 = split4;}


    if(currentplayer > 1){hit(s, currentplayer, p , cc);}
    if(currentplayer > 1){cout << "..............................Playing hand " << hand[currentplayer] << " ...........................\n"; }
    if(currentplayer > 1){printArray (p, hand[currentplayer],currentplayer , bet,  w);}
    if(currentplayer == 1){natural[currentplayer] = checkNatural (p, house, bet, currentplayer);}
        if(natural[1] < 0){bank -= natural[currentplayer]; cout << "Your total is now " << bank << " chips \n"; currentplayer = totalplayers + 1;}
        if(natural[1] > 1){bank += natural[currentplayer]; cout << "Your total is now " << bank << " chips \n"; currentplayer = totalplayers + 1;}
        if(natural[1] > 1){currentplayer = currentplayer + 1; break;}
        if(currentplayer > 1){natural[currentplayer] = checkNatural (p, house, bet, currentplayer);}
    while(splittime[0] == 4)
    {
            if(totalplayers == 1){p1 = split1;}
            if(totalplayers == 2){p1 = split2;}
            if(totalplayers == 3){p1 = split3;}
            if(totalplayers == 4){p1 = split4;}
        CheckSplit(s, p, p1, cc, bet, 4);
            if(splittime[0] == 4 && totalplayers == 5){cout << "you have slit your hand...............................................................\n";}
            if(splittime[0] == 4 && totalplayers == 5){printArray (split3, "Split 3 ", 4, bet,  w);}
            if(splittime[0] == 4 && totalplayers == 5){printArray (split4, "Split 4 ", 5, bet,  w);}
            cout << "......................................................................................\n";
        }

        while(splittime[0] == 3)
        {
            if(totalplayers == 1){p1 = split1;}
            if(totalplayers == 2){p1 = split2;}
            if(totalplayers == 3){p1 = split3;}
            if(totalplayers == 4){p1 = split4;}
            CheckSplit(s, p, p1, cc, bet, 3);
            if(splittime[0] == 3 && totalplayers == 4){cout << "you have slit your hand...............................................................\n";}
            if(splittime[0] == 3 && totalplayers == 4){printArray (split2, "Split 2 ", 3, bet,  w);}
            if(splittime[0] == 3 && totalplayers == 4){printArray (split3, "Split 3 ", 4, bet,  w);}
            if(splittime[0] == 3 && totalplayers == 5){printArray (split4, "Split 4 ", 5, bet,  w);}
            cout << "......................................................................................\n";
        }

        while(splittime[0] == 2)
        {
            if(totalplayers == 1){p1 = split1;}
            if(totalplayers == 2){p1 = split2;}
            if(totalplayers == 3){p1 = split3;}
            if(totalplayers == 4){p1 = split4;}
            CheckSplit(s, p, p1, cc, bet, 2);
            if(splittime[0] == 2 && totalplayers == 3 ){cout << "you have slit your hand...............................................................\n";}
            if(splittime[0] == 2 && totalplayers == 3 ){printArray (split1, "Split 1 ", 2, bet,  w);}
            if(splittime[0] == 2 && totalplayers == 3 ){printArray (split2, "Split 2 ", 3, bet,  w);}
            if(splittime[0] == 2 && totalplayers == 4 ){printArray (split3, "Split 3 ", 4, bet,  w);}
            if(splittime[0] == 2 && totalplayers == 5 ){printArray (split4, "Split 4 ", 5, bet,  w);}
            cout << "......................................................................................\n";
        }

        while(splittime[0] == 1)
        {
            if(totalplayers == 1){p1 = split1;}
            if(totalplayers == 2){p1 = split2;}
            if(totalplayers == 3){p1 = split3;}
            if(totalplayers == 4){p1 = split4;}
            CheckSplit(s, p, p1, cc, bet, 1);

            if(splittime[0] == 1 && totalplayers == 2 ){cout << "you have slit your hand...............................................................\n";}
        if(splittime[0] == 1 && totalplayers == 2 ){printArray (player, "Original  ", 1 , bet,  w);}
            if(splittime[0] == 1 && totalplayers == 2){printArray (split1, "Split 1 ", 2, bet,  w);}
            if(splittime[0] == 1 && totalplayers == 3){printArray (split2, "Split 2 ", 3, bet,  w);}
            if(splittime[0] == 1 && totalplayers == 4){printArray (split3, "Split 3 ", 4, bet,  w);}
            if(splittime[0] == 1 && totalplayers == 5){printArray (split4, "Split 4 ", 5, bet,  w);}
            cout << ".....................................................................................
```

```
.\n";
        }

        int now = currentplayer;
    while(now == currentplayer)
    {
        if(cardValue(p) >= 21){resp = 's';
    }else if(cardValue(p) < 21){cout << "What would you like to do for your "<< hand[currentplayer] << " hand (? for help)?\n" ;}
                    while(!(cin >> resp))
                             {
                                 cout <<"Invalid data type! Please enter 'y' or 'n'";
                                 cin.clear();
                                 cin.ignore(999,'\n');
                             }

        if (resp == 'h' || resp == 'H'){
                hit(s, currentplayer, p , cc);
                printArray (p, hand[currentplayer],currentplayer , bet,  w);
            } else if (resp == 'd' || resp == 'D'){
                cout << "Doubling down.....................................................\n";
                doubleDown(s, currentplayer, p, cc, bet);
                printArray (p, hand[currentplayer],currentplayer , bet,  w);
                now -= now;
            } else if (resp == 's' || resp == 'S')
                { now -= now; cout << "Staying.....................................................\n";
            } else if (resp == 'p' || resp == 'P')
                { printArray (p, hand[currentplayer],currentplayer , bet,  w);
            } else if (resp == 'b' || resp == 'B')
                { cout << "Chips have the value of $10.00, How many chips would you like to Buy? \n";
                while(!(cin >> bank))
                             {
                                 cout<<"Invalid data type! Please enter 'a number'";
                                 cin.clear();
                                 cin.ignore(999,'\n');
                             }
            } else if (resp == '?' || resp == '/')
                                {cout << "key commands you can use. h = hit, s = stay, d = double down, p = print hand, b = buy chips ? = Help   \n";

            }else {cout << "bubba \n";}
         if(cardValue(p) >= 21){now -= now;}

    }
        currentplayer++;
    }

    if(split1[0] !=0 ){ printArray (split1, "Split 1 ", 2, bet,  w);}
    if(split2[0] !=0){ printArray (split2, "Split 2 ", 3, bet,  w);}
    if(split3[0] !=0){ printArray (split3, "Split 3 ", 4, bet,  w);}
    if(split4[0] !=0){ printArray (split4, "Split 4 ", 5, bet,  w);}

    int checkhit = 1;
    if(natural[1] > 1 && currentplayer >= 2){checkhit = 0;}
    while(checkhit !=0)
    {
        cout << "The house is thinking about hitting or not..............................\n";
        scores[1] = cardValue(player);
        if(split1[0] != 0 ){ scores[2] = cardValue(split1);}
        if(split2[0] != 0 ){ scores[3] = cardValue(split1);}
        if(split3[0] != 0 ){ scores[4] = cardValue(split1);}
        if(split4[0] != 0 ){ scores[5] = cardValue(split1);}


        if(totalplayers == 1 && scores[1] > 21){dealerhit=0;
        }else
        {
        for(int i = 0; i < (totalplayers + 1); i++){if(scores[i] <= 21){dealerhit++;}}
        if(dealerhit >= 1){hit(s, 0, house, cc);;}
        if(totalplayers >= 1 ){printArray (house, "dealer  ",0, bet, w);}
        if(totalplayers >= 1 ){printArray (player, "Original  ",1 , bet,  w);}
        if(totalplayers >= 2 ){printArray (split1, "Split 1 ", 2, bet,  w);}
        if(totalplayers >= 3 ){printArray (split2, "Split 2 ", 2, bet,  w);}
        if(totalplayers >= 4 ){printArray (split3, "Split 3 ", 2, bet,  w);}
        if(totalplayers >= 5 ){printArray (split4, "Split 4 ", 2, bet,  w);}
        }
        scores[0] = cardValue(house);
        scores[1] = cardValue(player);
        scores[2] = cardValue(split1);
        scores[3] = cardValue(split2);
        scores[4] = cardValue(split3);
        scores[5] = cardValue(split4);
        bank = winner(scores, bank, bet, natural);
        checkhit--;
    }

    if(cardT <= reshuffling)
    {
    newDeck (x, arr);
    randomize (arr, n);
    randomize (arr, n);
    randomize (arr, n);
    randomize (arr, n);
    cardT = x;
    cout << "Cut cards, pick a number between 1 and " << x << ": \n";
    while(!(cin >> c))
             {
                 cout<<"Invalid data type! Please enter 'a number' between 1 and " <<
```

```
x;
                 cin.clear();
                 cin.ignore(999,'\n');
             }
    if(c < 1 || c > x){c = 1;}
    cutCards(arr, x, c);
    fillStack(s, arr, x);
    srand (time(NULL)); /* initialize random seed: */
    reshuffling = rand() % 25 + 26;/* generate a number between 25 and 50: */
    cout << "The Card Shoe is filled and the reshuffling card has been placed.\n\n";
    }
    if(bump > 0){cout << "you bumped this hand \n"; bump--;}
    if(bump == 0){ cout << "You have " << bank << " chips.  Please place your bet:";
    while(!(cin >> bet[1]))
     {
         cout<<"Invalid data type! Please enter 'a number'";
         cin.clear();
         cin.ignore(999,'\n');
     }
    }
}

    return 0;
    }

STACK *init(int n)
{
   STACK *s = (STACK *)malloc(sizeof(STACK));
   s->top = 0;
   s->maxelem = n;
   s->stk = (int *)malloc(sizeof(int)*n);
   return s;
}

void fillStack(STACK *s, int arr[], int x)
{
    for(int i = 0; i < x; i++){
        //cout << "top " << top << "\n";
        //cout << "I " << i << "\n";
        s->stk[s->top++] = arr[i];
  }
}

void push(STACK *s, int data)
{
   if (s == NULL) return;
   if (s->top == s->maxelem) return;
   s->stk[s->top++] = data;
}

int pop(STACK *s)
```

```
{
   if (s == NULL) return -1;
   if (s->top == 0) return -1;
   s->top--;
   return  s->stk[s->top];
}

void deleteNode(STACK *s, int data)
{
   if (s == NULL) return;
   if (s->stk[s->top-1] == data) { s->top--; return; }
   for (int i=s->top-1 ; i >= 0 ; i--)
   {
      if (s->stk[i] == data)
      {
         for ( ; i < s->top ; i++) s->stk[i] = s->stk[i+1];
         s->top--;
         return;
      }
   }
}

void doubleDown(STACK *s, int pno, int a[], int count[pno], double bet[pno])
{

    int b = count[pno];
    a[b] = pop(s);
    count[pno]++;
    bet[pno] =  bet[pno] * 2;
    cardT--;
}

void hit(STACK *s, int pno, int a[], int count[pno])
{
        int b = count[pno];
        int val = cardValue(a);
        if(pno == 0)
    {

        while(val <= 17)
        {
            a[b] = pop(s);
            cardT--;
            count[pno]++;
            cout << "dealer takes card number " << count[pno] << " ";
            b = count[pno];
            val = cardValue(a);
        }
        cout << "\n";
        if(val <= 21 ){cout << "Dealer Stands..................................\n";}
        if(val > 21 ){cout << "Dealer is bust................................\n";}
```

```cpp
    }
        if(pno != 0)
        {
        a[b] = pop(s);
        count[pno]++;
        cardT--;
        }
}

void dealCards(STACK *s, int player[], int house[], int count[])
{
    for (int i = 0; i < 2; i++)
    {
            player[i] = pop(s);
            count[1]++;
            cardT--;
            house[i] = pop(s);
            count[0]++;
            cardT--;
      }
  }

void CheckSplit (STACK *s, int a[], int b[], int count[totalplayers], double bet[], int splitplayer)
{
    string ohand = "empty  ";
    string nhand = "empty  ";
    int split = 0;
    int spno = splitplayer;
    int spto = 0;

    if(splitplayer == 1){ohand = "'original' ";}
    if(splitplayer == 2){ohand = "'Split 1' ";}
    if(splitplayer == 3){ohand = "'Split 2' ";}
    if(splitplayer == 4){ohand = "'Split 3' ";}
    if(splitplayer == 5){ohand = "'Split 4' ";}
    int i = 0;

            if((a[0]%13)==(a[1]%13)){split = 1;}
            if(totalplayers == 1){spto = 2; nhand = "'Split 1' ";
            }else if(totalplayers == 2){spto = 3; nhand = "'Split 2' ";
            }else if(totalplayers == 3){spto = 4; nhand = "'Split 3' ";
            }else if(totalplayers == 4){spto = 5; nhand = "'Split 4' ";}

                if(split == 0){splittime[0]++; i++;}
                if(split == 1)
                    {
                        char answer;
                        cout << "would you like to split your " << ohand << "hand? ";
                        while(!(cin >> answer))
                              {
                                cout<<"Invalid data type! Please enter 'y' or 'n'";
                                cin.clear();
                                cin.ignore(999,'\n');
                              }

                      if( answer == 'y')
                          {

                              splittime[spto]++;
                              b[0] = a[1];
                              a[1] = 0;
                              totalplayers++;
                              count[spno]--;
                              count[spto]++;
                              bet[spto] = bet[spno];
                              hit(s, spno, a, count); //void hit(STACK *s, int pno, int a[], int count[pno])
                              cardT--;
                          }
                    }
  }

string cardName(int a[])
{
        string ret;
        string retb;
        int num = 0;

        for(int i = 0; a[i] != 0; i++){num++;}

            for(int i = 0; i < num; i++)
                            {
        int b = ((a[i] % 13));
        int y = (a[i] % 52);
                if (b == 1){ret =  "  Ace";}
                if (b == 2){ret =  "    2";}
                if (b == 3){ret =  "    3";}
                if (b == 4){ret =  "    4";}
                if (b == 5){ret =  "    5";}
                if (b == 6){ret =  "    6";}
                if (b == 7){ret =  "    7";}
                if (b == 8){ret =  "    8";}
                if (b == 9){ret =  "    9";}
                if (b == 10){ret = "   10";}
                if (b == 11){ret = " Jack";}
                if (b == 12){ret = "Queen";}
                if (b == 0){ret =  " King";}
                if ((y / 14) == 0){ret += "♦";}
                if (y / 14 == 1){ret += "♠";}
                if (y / 14 == 2){ret += "♥";}
                if (y / 14 == 3){ret += "♣";}
                if (i < num - 1){ret = ret + ", ";}
                if(num >= 1){retb += ret;}
                else{ret = ", " + retb;}
                }
        return retb;
}

string cardNamed(int a)
{
            string ret;

            int b = ((a % 13));
            int y = (a % 52);
            if (b == 1){ret =  "  Ace";}
            if (b == 2){ret =  "    2";}
            if (b == 3){ret =  "    3";}
            if (b == 4){ret =  "    4";}
            if (b == 5){ret =  "    5";}
            if (b == 6){ret =  "    6";}
            if (b == 7){ret =  "    7";}
            if (b == 8){ret =  "    8";}
            if (b == 9){ret =  "    9";}
            if (b == 10){ret = "   10";}
            if (b == 11){ret = " Jack";}
            if (b == 12){ret = "Queen";}
            if (b == 0){ret =  " King";}
            if ((y / 14) == 0){ret += "♦";}
            if (y / 14 == 1){ret += "♠";}
            if (y / 14 == 2){ret += "♥";}
            if (y / 14 == 3){ret += "♣";}

    return ret;
}

int cardDealer(int a)
{
              int ret = 0;
              int aceflag = 0;
              int b = (a % 13);
                if (b > 10){ret += 10;}
                if (b == 0){ret += 10;}
                if (b == 1){ret += 11; ++aceflag;}
                if ((b > 1) && (b <= 10)){ret = ret +b;}

            return ret ;
}


```

```
int cardValue(int a[])
     {
       int ret = 0;
       int aceflag = 0;
        for(int i = 0; a[i] != 0; i++)
            {
            int b = ((a[i]) % 13);
            if (b > 10){ret += 10;}
            if (b == 0){ret += 10;}
            if (b == 1){ret += 11; ++aceflag;}
            if ((b > 1) && (b <= 10)){ret = ret +b;}
            if ((ret > 21) && (aceflag !=0 )){ret -= 10; --aceflag;}
            }
        return ret ;
}

int winner(int a[], int bank, double bet[], int natural[])
{       int i = 1;
    while (i <= totalplayers)
       {
           string hand[6]={"dealer", "Original  ", "Split 1 ", "Split 2 ", "Split 3 ", "Split 4 "};
           // natural 21
           if(natural[i] >= 1)
               {
               bank += (natural[i]); cout << "BlackJack Natural!! ......." << "you win your bet(" << natural[i] << ").  Your bank is now " << bank << " chips....\n";
               }

           // dealer is bust
           if(a[0] > 21)
                 {
                     if(a[i] <= 21){bank += (bet[i]); cout << hand[i] << " wins ......." << "you win your bet(" << bet[i] << ").  Your bank is now " << bank << " chips....\n";}
                     cout << "\n";
                 }

           //player bumps
             if(a[i] !=0 && a[0] <=21 && a[i] <= 21 && a[0] == a[i])
             {
                cout << hand[i] << " bumps ......." << hand[i] << ", your bet(" << (bet[i]) << ") will be put back.  Your bank is now " << bank << " chips..... \n";
                cout << "\n";
             }

          // player is bust
               if(a[i] !=0 && a[i] > 21)
                       {
                       bank -= bet[i];
                       cout << hand[i] << " you are bust, you loose your bet(" <<
bet[i] << ") your bank is now " << bank << " chips..... \n";
           cout << "\n";
           }

     // Dealer wins player bet
      if(a[i] !=0 && a[0] <= 21 && a[0] > a[i] && a[i] < 21)
       {
         bank -= (bet[i]);
         cout << hand[i] << " you lost, " << hand[0] << " wins, you loose your bet(" << (bet[i]) << ").  Your bank is now " << bank << " chips....\n";
         cout << "\n";
       }

     //player wins bet
      if(a[i] !=0 && a[i] <=21 && a[0] <= 21 && a[0] < a[i])
         {
          bank += (bet[i]);
          cout << hand[i] << " wins ......." << "you win your bet(" << bet[i] << ").  Your bank is now " << bank << " chips....\n";
          cout << "\n";
         }
      i++;
   }
  return bank;
}

  // A function to generate a new ordered deck of cards in arr[]
void newDeck (int x, int arr[x])
{
      for (int i = 0; i < x; i++)
      arr[i] = i+1 ;
}

  // A utility function to print an array
void printArray (int cardArr[], string whoIs, int pno, double bet[pno], int n)
{
    switch(n){
    case 0:
    cout << whoIs <<"          " << cardNamed(cardArr[0])<< ", "<< "  downcard" <<  "\n";
    break;

    case 1:
      if(pno == 0){
        cout << whoIs << "            " << cardName(cardArr) << "  Total: " << cardValue(cardArr) <<  "\n";
      }
      if(pno >= 1){
        cout << whoIs <<"bet(" << bet[pno] << ")  " << cardName(cardArr) << " Total: " << cardValue(cardArr) << "\n";
      }
```

```
        break;
        }
        cout << "\n";
}

     // A utility function to swap to integers
void swap (int *a, int *b)
{
         int temp = *a;
         *a = *b;
         *b = temp;
}

     // A function to generate a random permutation of arr[]
void randomize ( int arr[], int n )
{
         srand ( time(NULL) );

         // Start from the last element and swap one by one.
         // We don't need to run for the first element that's why i > 0
         for (int i = n-1; i > 0; i--)
         {
         srand (time(NULL)); /* initialize random seed: */
             // Pick a random index from 0 to i
             int j = rand() % (i+1);

             // Swap arr[i] with the element at random index
            swap(&arr[i], &arr[j]);
         }
        // cout << "\n";
}

     // A utility function to swap to integers
void cutCards(int arr[], int x, int c)
{
        c-=1;
         int temp[x];
         int j = c;
        for(int i = 0; i < x; i++)
        {
            if (j >= x){j = (j) - x ;}

        temp[i] = arr[j];
        j++;

        }
            for(int i = 0; i < x; i++)
            {
                arr[i] = temp[i];
            }
}
```

```cpp

int checkNatural(int a[], int b[], double bet[], int pno)
{
     int ret = 0;
     if(cardValue(a) == 21)
          {
              int bjp = 0;
              int bjd = 0;
              int tens[2] = {0};
              int aceflag[2] = {0};
                  for(int i = 0; a[i] != 0; i++)
                  {
                      int b = ((a[i]) % 13);
                      if (b > 10){tens[1] = 1;}
                      if (b == 0){tens[1] = 1;}
                      if (b == 1){++aceflag[1];}
                  }
                  for(int i = 0; a[i] != 0; i++)
                  {
                      int c = ((b[i]) % 13);
                      if (c > 10){tens[0] = 1;}
                      if (c == 0){tens[0] = 1;}
                      if (c == 1){++aceflag[0];}
                  }
                  if(aceflag[1] == 1 && tens[1] == 1){bjp = 1;}
                  if(aceflag[0] == 1 && tens[0] == 1){bjd = 1;}
                  if(bjp == 0){ret = 0; cout << "no natural \n";}
                  if(bjp > bjd){ret = bet[pno] * 1.5; cout << "You have a 'natural' your bet is payed at 1.5 times its value! \n";}
                  if(bjp == bjd){ret = bet[pno]; cout << "You have a 'natural,' the dealer has a 'natural' your bet is bumped! \n";}
          }
       return ret;
   }

```