```cpp
// CrapsProject.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>
#include <string>
using namespace std;

enum Spot { FILLED, UNFILLED };

class Dice {
public:
    int roll() {
        return (rand() % (6 - 1 + 1)) + 1;
    }
};


class Positions {
public:
    bool contains(int value) {
        for (int i = 0; i < 5; i++)
        {
            if (positions[i] == value) {
                return true;
            }
        }
        return false;
    }

    void setPositionAt(int index, int value) {
        positions[index] = value;
    }

    void clear() {
        for (int i = 0; i < 5; i++)
        {
            setPositionAt(i, 0);
        }
    }

private:
    int positions[5] = { 0,0,0,0,0 };
};
```

```
49  class PlayerSpot {
50  public:
51      PlayerSpot() {
52          status = UNFILLED;
53          loseAmmount = 0;
54          winAmmount = 0;
55          betAmount = 0;
56          initialDiceRoll = 0;
57          name = "";
58      }
59
60      void clear() {
61          status = UNFILLED;
62          loseAmmount = 0;
63          winAmmount = 0;
64          betAmount = 0;
65          initialDiceRoll = 0;
66          name = "";
67      }
68
69
70      Spot getSpotStatus() {
71          return status;
72      }
73      void setSpotStatus(Spot definedStatus) {
74          status = definedStatus;
75      }
76      double getBetAmount() {
77          return betAmount;
78      }
79      void setBetAmount(double dollars) {
80          betAmount = dollars;
81      }
82      void setInitialDiceRoll(int roll) {
83          initialDiceRoll = roll;
84      }
85      int getInitialDiceRoll() {
86          return initialDiceRoll;
87      }
88      void setwinAmmount(double dollars) {
89          winAmmount = dollars;
90      }
91      double getwinAmmount() {
92          return winAmmount;
93
94      }
95      void setloseAmmount(double dollars) {
96          loseAmmount = dollars;
97      }
```

```
 98      double getloseAmmount() {
 99          return loseAmmount;
100      }
101      void setPlayerName(string playerName) {
102          name = playerName;
103      }
104      string getPlayerName() {
105          return name;
106      }
107
108
109
110  private:
111      Spot status;
112      double loseAmmount;
113      double winAmmount;
114      double  betAmount;
115      int initialDiceRoll;
116      int currentBetDiceRoll;
117      int numberOfPlayers;
118      string name;
119  };
120
121
122
123  int main()
124  {
125      srand(time(0));
126      int numberOfPlayers;
127      double betAmountInput;
128
129      Dice diceOne{};
130      Dice diceTwo{};
131
132      PlayerSpot playerSpot1, playerSpot2, playerSpot3, playerSpot4,
           playerSpot5;
133      PlayerSpot playerSpots[5] = { playerSpot1, playerSpot2, playerSpot3,
           playerSpot4, playerSpot5 };
134
135      Positions playerPositions{};
136
137
138
139
140      while (true)
141      {
142          cout << "--> Welcome back!\n";
143          cout << "Enter number of players: \n";
144          cin >> numberOfPlayers;
```

```
145         system("cls");
146         if (numberOfPlayers >= 2 && numberOfPlayers <= 5) {
147
148             //Get player bets
149             for (int i = 0; i < numberOfPlayers; ++i) {
150                 if (playerSpots[i].getSpotStatus() == UNFILLED) {
151
152                     playerSpots[i].setSpotStatus(FILLED);
153                     cout << "Player " << i + 1 << " how much are you
                    betting? \n";
154                     cin >> betAmountInput;
155                     playerSpots[i].setBetAmount(betAmountInput);
156                     playerSpots[i].setPlayerName("Player " + to_string(i +
                    1));
157                 }
158             }
159
160
161             //Get player positions
162             for (int i = 0; i < numberOfPlayers; i++)
163             {
164                 int orderRoll = diceOne.roll() + diceTwo.roll();
165                 while (playerPositions.contains(orderRoll)) {
166                     orderRoll = diceOne.roll() + diceTwo.roll();
167                 }
168                 playerPositions.setPositionAt(i, orderRoll);
169                 playerSpots[i].setInitialDiceRoll(orderRoll);
170             }
171
172             //Show inicial player positions
173             cout << "\n\n--> Initial Players positions\n";
174             for (int i = 0; i < numberOfPlayers; i++)
175             {
176                 cout << playerSpots[i].getPlayerName() << " has rolled
                  initialy: " << playerSpots[i].getInitialDiceRoll() <<
                  "\n";
177             }
178
179
180             //Order player positions
181             for (int i = 0; i < numberOfPlayers; i++)
182             {
183                 for (int j = 0; j < numberOfPlayers; j++)
184                 {
185                     if (playerSpots[j].getInitialDiceRoll() < playerSpots
                    [i].getInitialDiceRoll()) {
186                         PlayerSpot temp = playerSpots[i];
187                         playerSpots[i] = playerSpots[j];
188                         playerSpots[j] = temp;
```

```
189                         }
190                     }
191                 }
192                 cout << "\n\n--> Ordered Players positions\n";
193                 for (int i = 0; i < numberOfPlayers; i++)
194                 {
195                     cout << playerSpots[i].getPlayerName() << " has rolled
                      initialy: " << playerSpots[i].getInitialDiceRoll() <<
                      "\n";
196                 }
197
198                 bool continuePlaying = true;
199                 int lastRoll = 0;
200
201                 //Play craps
202                 cout << "\n--> Game Starts\n";
203                 while (continuePlaying)
204                 {
205                     for (int i = 0; i < numberOfPlayers; i++)
206                     {
207                         int currentBetRoll = diceOne.roll() + diceTwo.roll();
208                         cout << playerSpots[i].getPlayerName() << " rolls: "
                    << currentBetRoll << "\n";
209
210                         if (i == 0 && (currentBetRoll == 7 || currentBetRoll
                    == 11)) {
211                             double totalPayment = playerSpots[i].getBetAmount
                    () * (numberOfPlayers - 1);
212                             playerSpots[i].setwinAmmount(totalPayment);
213
214                             for (int j = 1; j < numberOfPlayers; j++)
215                             {
216                                 playerSpots[j].setloseAmmount(playerSpots
                    [i].getBetAmount());
217
218                             }
219                             cout << playerSpots[i].getPlayerName() << " wins
                    = )\n ";
220                             continuePlaying = false;
221                             break;
222                         }
223
224                         if (i == 0 && (currentBetRoll == 2 || currentBetRoll
                    == 3 || currentBetRoll == 12)) {
225                             double totalLost = 0;
226
227                             for (int j = 1; j < numberOfPlayers; j++)
228                             {
229                                 totalLost += playerSpots[j].getBetAmount();
```

```
230                             playerSpots[j].setwinAmmount(playerSpots
                    [j].getBetAmount());
231                     }
232                     cout << playerSpots[i].getPlayerName() << " loses
                    = (\n";
233                     playerSpots[i].setloseAmmount(totalLost);
234
235
236                     continuePlaying = false;
237                     break;
238                  }
239
240
241                  if (lastRoll == currentBetRoll) {
242                     double totalPayment = playerSpots[i].getBetAmount
                    () * (numberOfPlayers - 1);
243                     playerSpots[i].setwinAmmount(totalPayment);
244                     cout << playerSpots[i].getPlayerName() << " wins
                    =)\n";
245
246                     for (int j = 0; j < numberOfPlayers; j++)
247                     {
248                         if (j != i) {
249                             playerSpots[j].setloseAmmount(playerSpots
                    [i].getBetAmount());
250                         }
251                     }
252                     continuePlaying = false;
253                     break;
254                  }
255
256                  if (i != 0 && currentBetRoll == 7) {
257                     double totalLost = 0;
258                     cout << playerSpots[i].getPlayerName() << " lost =
                    (\n";
259
260                     for (int j = 0; j < numberOfPlayers; j++)
261                     {
262                         if (j != i) {
263                             totalLost += playerSpots[j].getBetAmount
                    ();
264                             playerSpots[j].setwinAmmount(playerSpots
                    [j].getBetAmount());
265                         }
266                     }
267                     playerSpots[i].setloseAmmount(totalLost);
268                     continuePlaying = false;
269                     break;
270                  }
```

```
271                     lastRoll = currentBetRoll;
272                 }
273             }
274
275             //Show win and lose ammounts
276             cout << "\n--> Final score and debts\n";
277             for (int i = 0; i < numberOfPlayers; i++)
278             {
279                 cout << playerSpots[i].getPlayerName() << " win ammount: "
                       << playerSpots[i].getwinAmmount() << "\n";
280                 cout << playerSpots[i].getPlayerName() << " lose ammount:
                      " << playerSpots[i].getloseAmmount() << "\n\n";
281             }
282             cout << "Thanks for playing, see you soon!\n\n";
283
284             playerPositions.clear();
285
286             for (int i = 0; i < numberOfPlayers; i++)
287             {
288                 playerSpots[i].clear();
289             }
290         }
291         else {
292             cout << "\nInvalid number of players \nPlayers must be between
                   2 and 5\n\n";
293         }
294
295     }
296     return 0;
297 }
298
```