```cpp
#include "Board.h"
#include "Slot.h"
#include <time.h>
#include <stdlib.h>
#include <iostream>
#include <iomanip>
using namespace std;

string center(int width, const string& str) {
    int len = str.length();
    if (width < len) { return str; }

    int diff = width - len;
    int pad1 = diff / 2;
    int pad2 = diff - pad1;
    return string(pad1, ' ') + str + string(pad2, ' ');
}

void draw_edge_line(int width, const string* line)
{
    cout << "|" << string((width) * 5 + 4, '-') << "|" << endl;
    cout << "|" << string((width) * 5 + 4, ' ') << "|" << endl;
    cout << "|";
    for (int col = 0; col < 5; col++)
        cout << center(width, line[col]) << "|";
    cout << endl;
    cout << "|" << string((width) * 5 + 4, ' ') << "|" << endl;
    cout << "|" << string((width) * 5 + 4, '-') << "|" << endl;
}

void draw_inner_line(int width, const string* line, bool last)
{
    cout << "|" << string(width * 5 + 4, ' ') << "|" << endl;
    cout << "|" << center(width, line[0]) << "|";
    cout << string(width * 3 + 2, ' ') << "|";
    cout << center(width, line[4]) << "|" << endl;
    cout << "|" << string(width * 5 + 4, ' ') << "|" << endl;
    if (!last)
    {
        cout << "|" << string(width, '-') << "|" << string((width * 3 + 3), '
          ');
        cout << string(width, '-') << "|" << endl;
    }
}


ostream& operator<<(ostream& os, const Board& b)
{
    draw_edge_line(b.m_slot_width, b.m_board_image[0]);
    for (int row = 1; row < 5; row++)
        draw_inner_line(b.m_slot_width, b.m_board_image[row], row == 4);
    draw_edge_line(b.m_slot_width, b.m_board_image[5]);
    return os;
}

void Board::init_board_image()
```

```
56  {
57      m_board_image[0][0] = m_arr[9]->get_name();
58      m_board_image[0][1] = m_arr[10]->get_name();
59      m_board_image[0][2] = m_arr[11]->get_name();
60      m_board_image[0][3] = m_arr[12]->get_name();
61      m_board_image[0][4] = m_arr[13]->get_name();
62      m_board_image[1][0] = m_arr[8]->get_name();
63      m_board_image[1][1] = "";
64      m_board_image[1][2] = "";
65      m_board_image[1][3] = "";
66      m_board_image[1][4] = m_arr[14]->get_name();
67      m_board_image[2][0] = m_arr[7]->get_name();
68      m_board_image[2][1] = "";
69      m_board_image[2][2] = "";
70      m_board_image[2][3] = "";
71      m_board_image[2][4] = m_arr[15]->get_name();
72      m_board_image[3][0] = m_arr[6]->get_name();
73      m_board_image[3][1] = "";
74      m_board_image[3][2] = "";
75      m_board_image[3][3] = "";
76      m_board_image[3][4] = m_arr[16]->get_name();
77      m_board_image[4][0] = m_arr[5]->get_name();
78      m_board_image[4][1] = "";
79      m_board_image[4][2] = "";
80      m_board_image[4][3] = "";
81      m_board_image[4][4] = m_arr[17]->get_name();
82      m_board_image[5][0] = m_arr[4]->get_name();
83      m_board_image[5][1] = m_arr[3]->get_name();
84      m_board_image[5][2] = m_arr[2]->get_name();
85      m_board_image[5][3] = m_arr[1]->get_name();
86      m_board_image[5][4] = m_arr[0]->get_name();
87
88      m_slot_width = 0;
89      for (int row = 0; row < 6; row++)
90      {
91          for (int col = 0; col < 5; col++)
92              if (m_board_image[row][col].size() > m_slot_width)
93                  m_slot_width = m_board_image[row][col].size();
94      }
95      m_slot_width += 2;
96  }
97
98  Board::Board()
99  {
100     srand(time(NULL));
101     m_size = 0;
102     add_go_slot("GO!");
103     add_asset_slot("Jerusalem", "zoo");
104     add_asset_slot("Jerusalem", "David_tower");
105     add_asset_slot("Jerusalem", "Western_wall");
106     add_jail_slot("JAIL! Wait 1 turn");
107
108     add_asset_slot("Tel_Aviv", "Hilton");
109     int num = rand() % 1000 + 500;
110     add_chance_slot("You won the lottery", num);
111     add_asset_slot("Tel_Aviv", "Azrieli");
```

```
112     add_asset_slot("Tel_Aviv", "Habima");
113     num = rand() % 200 + 100;
114     add_chance_slot("You have to pay the IRS", -num);
115
116     add_asset_slot("Carmiel", "Rocks_park");
117     num = rand() % 70 + 30;
118     add_chance_slot("Congratulations to your birthday, get a bouquet", num);
119     add_asset_slot("Carmiel", "Big");
120     add_asset_slot("Carmiel", "Ort_Braude");
121
122     add_asset_slot("Eilat", "Dolpin_reef");
123     add_asset_slot("Eilat", "Kings_town");
124     add_asset_slot("Eilat", "Ramon_airport");
125     add_asset_slot("Eilat", "Almog_beach");
126     init_board_image();
127 }
128
129 void Board::increase_board()
130 {
131     Slot ** tmp = new Slot *[m_size + 1];
132     int i;
133     for (i = 0; i < m_size; i++)
134         tmp[i] = m_arr[i];
135     m_size++;
136     if (m_arr)
137         delete[] m_arr;
138     m_arr = tmp;
139 }
140
141 void Board::add_asset_slot(const string& city, const string& asset_name)
142 {
143     increase_board();
144     m_arr[m_size-1] = new Asset(m_size, city, asset_name);
145 }
146
147
148 void Board::add_go_slot(const string& text)
149 {
150     increase_board();
151     m_arr[m_size - 1] = new Go(m_size, text);
152 }
153
154 void Board::add_jail_slot(const string& text)
155 {
156     increase_board();
157     m_arr[m_size - 1] = new Jail(m_size, text);
158 }
159
160 void Board::add_chance_slot(const string& text, float amount)
161 {
162     increase_board();
163     m_arr[m_size - 1] = new Chance(m_size, text, amount);
164 }
165
166 int Board::size() const
167 {
```

```
168     return m_size;
169 }
170
171 Slot * Board::operator[](int idx) const
172 {
173     return m_arr[idx];
174 }
175
176 istream& operator >> (istream& is, Board::action& i)
177 {
178     int tmp;
179     if (is >> tmp)
180         i = (Board::action)(tmp);
181     return is;
182 }
183
184 void Board::print_help()
185 {
186     cout << "\nto continue press (" << PLAY << "),";
187     cout << " To print board press(" << PRINT_BOARD << "),";
188     cout << " To end game press(" << END_GAME << ")\n";
189 }
190
191 Board::action Board::get_command() const
192 {
193     Board::action cmd;
194     cin >> cmd;
195     if (cin.fail() || cmd < 0 || cmd > 2)
196     {
197         cin.clear();
198         cin.ignore();
199         return get_command();
200     }
201     return cmd;
202 }
203
204 void Board::play(Player* players)
205 {
206     int player = 0;
207     action a;
208     while (1)
209     {
210         cout << players[player].get_name() << "'s turn: ";
211         print_help();
212         a = (action)get_command();
213
214         if (a == END_GAME)
215             break;
216         else if (a == PRINT_BOARD)
217         {
218             cout << *this;
219             continue;
220         }
221         else if (a == PLAY)
222         {
223             if (!(players[player]).draw_dice())
```

```
224                 break;
225             cout << players[player];
226             player = (player + 1) % Player::get_counter();
227         }
228     }
229     cout << "End of Game! Bye!" << endl;
230 }
231
```