```cpp
#include "map.h"
#include <cstdlib>
#include <iostream>
#include "utilities.h"

using namespace std;

bool _checkIfLakeCanBeAdd(Map& map, size_t originX, size_t originY, size_t radius);
void _addLake(Map& map, size_t originX, size_t originY, size_t radius);

bool _checkIfLakeCanBeAdd(Map& map, size_t originX, size_t originY, size_t radius) {
    size_t mapHeight = getHeight(map) - 1;
    size_t mapWidth  = getWidth(map) - 1;

    size_t minHeight = ((int)originY - (int)radius) > 0 ? ((int)originY - (int)radius) : 0;
    size_t maxHeight = (originY + radius) > mapHeight ? mapHeight : (originY + radius);


    for (size_t height = minHeight; height <= maxHeight; ++height) {
        size_t lineHalfWidth = radius - abs((int)originY - (int)height);
        size_t minWidth = ((int)originX - (int)lineHalfWidth) > 0
                        ? ((int)originX - (int)lineHalfWidth)
                        : 0;
        size_t maxWidth = (originX + lineHalfWidth) > mapWidth
                        ? mapWidth
                        : (originX + lineHalfWidth);

        for (size_t width = minWidth; width <= maxWidth; ++width) {
            if (getMapValue(map, height, width) != MS_EARTH) return false;
        }
    }
    return true;
}

void _addLake(Map& map, size_t originX, size_t originY, size_t radius) {
    size_t mapHeight = getHeight(map) - 1;
    size_t mapWidth  = getWidth(map) - 1;

    size_t minHeight = ((int)originY - (int)radius) > 0 ? ((int)originY - (int)radius) : 0;
    size_t maxHeight = (originY + radius) > mapHeight ? mapHeight : (originY + radius);

    for (size_t height = minHeight; height <= maxHeight; ++height) {
        size_t lineHalfWidth = radius - abs((int)originY - (int)height);
        size_t minWidth = ((int)originX - (int)lineHalfWidth) > 0
                        ? ((int)originX - (int)lineHalfWidth)
                        : 0;
        size_t maxWidth = (originX + lineHalfWidth) > mapWidth
                        ? mapWidth
                        : (originX + lineHalfWidth);

        for (size_t width = minWidth; width <= maxWidth; ++width) {
            setMapValue(map, height, width, MS_WATER);
        }
    }
}

size_t getHeight(const Map& map) {
   return map.size();
}

size_t getWidth(const Map& map) {
   if (map.empty()) return 0;
   return map[0].size();
}

Map getEmptyMap(size_t height, size_t width) {
   return Map(height, Axe(width, MS_EARTH));
}

MapState getMapValue(const Map& map, size_t x, size_t y) {
   if (y >= getHeight(map) or x >= getWidth(map))
      return MS_OUT;
   return map[y][x];
}

bool setMapValue(Map& map, size_t x, size_t y, MapState value) {
   if (y >= getHeight(map) or x >= getWidth(map))
```

```
78          return false;
79      map[y][x] = value;
80      return true;
81  }
82
83  bool addTreasure(Map& map, size_t x, size_t y) {
84      if (y >= getHeight(map) or x >= getWidth(map))
85          return false;
86      if (getMapValue(map, x, y) != MS_EARTH)
87          return false;
88      setMapValue(map, x, y, MS_TREASURE);
89      return true;
90  }
91
92  void addRandomTreasure(Map& map) {
93      size_t x;
94      size_t y;
95      do {
96          x = getRandomInRange(getHeight(map));
97          y = getRandomInRange(getWidth(map));
98      } while (!addTreasure(map, x, y));
99  }
100
101 // NB: if lake are added first, we could get their origins and radius
102 // if | origin1 - origin2 | > radius1 + radius2, then, they don't touch each other
103
104 bool addLake(Map& map, size_t originX, size_t originY, size_t radius) {
105     if (!_checkIfLakeCanBeAdd(map, originX, originY, radius)) return false;
106     _addLake(map, originX, originY, radius);
107     return true;
108 }
109
110 void addRandomLake(Map& map) {
111     int maxRadius = (getHeight(map) > getWidth(map) ? getWidth(map) : getHeight(map))
112                     / NUMBER_OF_LAKE;
113     addRandomLake(map, maxRadius);
114 }
115
116 void addRandomLake(Map& map, size_t maxRadius) {
117     size_t radius;
118     size_t height;
119     size_t width;
120     do {
121         radius = getRandomInRange(maxRadius);
122         height = getRandomInRange(getHeight(map));
123         width = getRandomInRange(getWidth(map));
124         #ifdef DEBUG
125         cout << "addRandomLake called" << endl;
126         cout << "maxRadius: " << maxRadius << endl;
127         cout << "radius: " << radius << endl;
128         cout << "height: " << height << endl;
129         cout << "width: " << width << endl;
130         #endif
131     } while (!addLake(map, height, width, radius));
132 }
133
134 bool addStart(Map& map, size_t x, size_t y) {
135     if (x >= getHeight(map) or y >= getWidth(map))
136         return false;
137     if (getMapValue(map, x, y) != MS_EARTH)
138         return false;
139     setMapValue(map, x, y, MS_START);
140     return true;
141 }
142
143 void addRandomStart(Map& map) {
144     size_t x;
145     size_t y;
146     addRandomStart(map, x, y);
147 }
148
149 void addRandomStart(Map& map, size_t& x, size_t& y) {
150     do {
151         x = getRandomInRange(getWidth(map));
152         y = getRandomInRange(getHeight(map));
153     } while (!addStart(map, x, y));
154 }
```

```
155
156  Map initWorld(size_t height, size_t width, size_t& startX, size_t& startY) {
157
158     Map map = getEmptyMap(height, width);
159
160     for (int i = 0; i < NUMBER_OF_LAKE; ++i) {
161        addRandomLake(map);
162     }
163     addRandomTreasure(map);
164     addRandomStart(map, startX, startY);
165
166     return map;
167  }
168
169
170  void displayWorld(const Map& map) {
171      for(const auto& axe : map) {
172          for(const auto state: axe) {
173              #ifdef _WIN32
174                  switch (state)
175                  {
176                  case MS_TREASURE:
177                      cout << "T";
178                      break;
179
180                  case MS_WATER:
181                      cout << "W";
182                      break;
183
184                  case MS_START:
185                      cout << "S";
186                      break;
187
188                  case MS_EARTH:
189                  default:
190                      cout << " ";
191                      break;
192                  }
193              #else
194                  switch (state)
195                  {
196                  case MS_TREASURE:
197                      cout << "\e[30;43m \e[0m";
198                      break;
199
200                  case MS_WATER:
201                      cout << "\e[30;44m \e[0m";
202                      break;
203
204                  case MS_START:
205                      cout << "\e[30;41m \e[0m";
206                      break;
207
208                  case MS_EARTH:
209                  default:
210                      cout << "\e[30;42m \e[0m";
211                      break;
212                  }
213              #endif
214          }
215          cout << endl;
216      }
217  }
```