P Open in editor

```
1 #include <bits/stdc++.h>
 2
 3 using namespace std;
 4
 5 string ltrim(const string &);
 6 string rtrim(const string &);
 7 vector<string> split(const string &);
9 int hourglassSum(vector<vector<int>> arr) {
10
       int max_sum = INT_MIN;
11
       for (int i = 0; i <= 3; i++) {
           for (int j = 0; j <= 3; j++) {
12
13
               int sum = arr[i][j] + arr[i][j + 1] + arr[i][j + 2] +
14
                                     arr[i + 1][j + 1] +
15
                          arr[i + 2][j] + arr[i + 2][j + 1] + arr[i + 2][j + 2];
16
               if (sum > max_sum) {
17
                   max_sum = sum;
18
               }
19
           }
20
21
       return max_sum;
22 }
23
24 int main()
25 {
26
       ofstream fout(getenv("OUTPUT_PATH"));
27
28
       vector<vector<int>> arr(6);
29
30
       for (int i = 0; i < 6; i++) {
31
           arr[i].resize(6);
32
33
           string arr_row_temp_temp;
34
           getline(cin, arr_row_temp_temp);
           vector<string> arr_row_temp = split(rtrim(arr_row_temp_temp));
36
37
38
           for (int j = 0; j < 6; j++) {
39
               int arr_row_item = stoi(arr_row_temp[j]);
40
41
               arr[i][j] = arr_row_item;
42
           }
43
       }
44
45
       int result = hourglassSum(arr);
46
47
       fout << result << "\n";
48
49
       fout.close();
50
51
       return 0;
52 }
53
54 string ltrim(const string &str) {
55
       string s(str);
56
57
       s.erase(
58
           s.begin(),
59
           find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
60
       );
61
62
       return s;
63 }
```

```
00 ]
64
65 string rtrim(const string &str) {
66
       string s(str);
67
68
       s.erase(
69
           find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
70
           s.end()
71
       );
72
73
       return s;
74 }
75
76 vector<string> split(const string &str) {
77
       vector<string> tokens;
78
79
       string::size_type start = 0;
80
       string::size_type end = 0;
81
       while ((end = str.find(" ", start)) != string::npos) {
82
83
           tokens.push_back(str.substr(start, end - start));
84
           start = end + 1;
85
       }
86
87
       tokens.push_back(str.substr(start));
88
89
       return tokens;
90 }
```