```
**********************************************************************************/
#include <bits/stdc++.h>
using namespace std;

int main()
{
    cout << "RADHE RADHE RAM RAM" << endl;

    auto ComputePrefix_same_as_Suffix = [&](string &s, vector <int> &PI) {
        int n = s.length();
        PI.resize(n, 0);

        int k = 0, cnt = 0;
        for(int q=2;q<=n;q++) {
            bool t = s[k+1] != s[q];
            cnt++;

            while(k>0 and t) {
                k = PI[k];
                t = s[k+1] != s[q];
                cnt++;
            }

            if(!t) {
                k++;
            }

            PI[q] = k;
        }
        cout << "PI table: [";
        for(auto &i: PI) {
            cout << i << ", ";
        }
        cout << "] " << cnt << '\n';
        // cout <<
    };

    // string s = "aaaaa";// -> 4 : [ 0, 1, 2, 3, 4 ]
    // string s = "acaca";// 4
    // string s = "abcde"; //4
    string s = "0AAACAAAAAC";
    // string s = "0aaabb";
    vector <int> PI;
    ComputePrefix_same_as_Suffix(s, PI);
```

```
    // auto KMP = [&](string &s, vector<int> &PI) {
    //     int n =
    // }



    return 0;
}

*******************************************************************************/

#include <bits/stdc++.h>
using namespace std;

#define f first
#define s second

int main()
{
    cout << "RADHE RADHe RAM RAM" << endl;

    vector <pair <int, int>> Points;

    Points.push_back({2,2});
    Points.push_back({4,4});
    Points.push_back({5,6});

    auto square = [&](int x)
    {
        return x * x;
    };

    auto closestPairPoint = [&]()
    {
        int n = Points.size();
        int minDist=  INT_MAX;
        pair <int, int> pairInd  = {-1, -1};

        for(int i=0;i<n-1;i++) {
            for(int j=i+1;j<n;j++) {
                int tempDist = square(Points[i].f - Points[j].f) + square(Points[i].s - Points[j].s);

                if(tempDist < minDist) {
```

```
                pairInd = {i, j};
                minDist = tempDist;
            }
        }
    }

    return pairInd;
};

pair <int, int> indices = closestPairPoint();

cout << indices.f+1 << ' ' << indices.s+1;
}
```