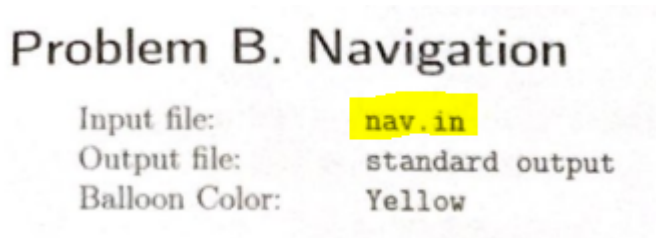
Here is a typical starter code for any problem in ECPC:

```cpp
#include <bits/stdc++.h>

using namespace std;


void solve(){
    // here write the code to solve a single test case

    // Don't forget to clear global variables between test cases
}

int32_t main() {
    freopen("file.in", "r", stdin);

    ios::sync_with_stdio( sync: 0);
    cin.tie( tiestr: 0);
    cout.tie( tiestr: 0);
    cout.precision( prec: 17);

    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- You need line 13 as the input for the problems are from a file and not STDIN. You need to include this line and change **“file.in”** with the file name stated in the problem



Problem B. Navigation

Input file: nav.in
Output file: standard output
Balloon Color: Yellow

- You need lines 15-17 to be able to use **cin** and **cout** without getting a **TLE**.
- Line 18 to print doubles with the highest possible precision.

Sometimes, you need to clear global variables between test cases. **Note that you can't Clear up to N in every test case. You just clear up to n which is the space needed for this test case.**

```
#define N 100009  // largest possible value + 9

vector<vector<int>> adjList(N);

void solve(){
    int n, m;
    cin >> n >> m;

    // Don't forget to clear global variables between test cases
    for(int i=0 ; i< n+5 ; i++)
        adjList[i].clear();
    // or
    //adjList.clear();
    //adjList.resize(n + 5);
}
```

If you didn't clear here you may find neighbours for the node from previous test cases.

One common Bug is overflow. One way to avoid it (without looking for it) is to do this:

```
#define int long long

void solve(){

}

int32_t main() {
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Now any **int** you use is converted to **long long.**
You will need to change “**int main”** to “**int32_t main”**