第一题：

梯度一：

1. 机甲⼤师系列赛由⾼校联盟赛，超级对抗赛分区赛和全国赛组成，其中⾼校联盟赛的时间是每年的 \_\_3月到4月\_\_\_\_(精确到⽉，下同）。超级对抗赛的分区赛时间是每年的\_\_\_5月到6月\_\_\_，共设有\_\_\_4\_\_\_个站点。 超级对抗赛全国赛的时间是每年的\_\_\_7月到8月\_\_\_，⽐赛地点在\_\_深圳湾体育中心春茧体育馆\_\_\_\_（精确到城市及体育馆）。

2. 机甲⼤师中有两种弹丸分别是17mm荧光⼩弹丸（下简称⼩弹丸）和42mm⼤弹丸（下简称⼤弹 丸），其中⼩弹丸对敌⽅机器⼈的伤害是\_\_\_\_2点\_\_，⼤弹丸对敌⽅机器⼈的伤害是\_\_\_4点\_\_\_。

3. ⻜镖是⽐赛中双⽅的⼤杀器，是双⽅的核威慑武器。Helios的⻜镖组在2025赛季取得了极⼤的进 展，其中在对战A战队的⼀⼩局⽐赛中在两轮⻜镖发射窗⼝发射了两次⻜镖其中第⼀个发射窗⼝向 敌⽅基地以随机固定⽬标发射了2发⻜镖，可惜只命中了⼀发，第⼆个发射窗⼝向敌⽅基地以随机 移动⽬标发射了2发⻜镖且全部命中⽬标，不禁让解说席的解说员激动的⾼呼“哇，两发全中” 则本局⻜镖系统共造成了\_\_\_2000\_\_\_的伤害并且使得敌⽅全体致盲了\_\_\_5\_\_\_秒，还造成了\_\_\_前哨站被击毁\_\_\_和 \_\_\_哨兵未阵亡\_\_\_的效果。

4. ⽼⻨是Helios的⼀辆传奇步兵，以极⾼的稳定性在今年的分区赛中⼤放异彩，⼤杀特杀，在⽐赛的 第四分钟，10级的⽼⻨成功激活了40环的能量机关，并在10s后成功通过⻜坡去准备击杀对⽅⾯的 哨兵机器⼈，假设完成⽼⻨的⾃瞄系统⼗分的鲁棒能做到命中率50%且机械结构⼗分完美射频能达 到最⾼40hz，在攻击哨兵是开启了雷达双倍的易伤，在⾃⾝不扣除⾎量的情况下⽼⻨需要多久才可 以击杀敌⽅的哨兵\_\_\_2.5s\_\_\_

梯度⼆：简答题 在机甲⼤师研发⾃⼰的机器⼈中他们遇到过许多困难，也取得了很多的成果，下⾯是⼀些经常挂在机 甲⼤师嘴边的词语，你都了解过这些是啥样⼦的意思吗： 镖群 君瞄 ⽅瞄 救命啊 整⻋预测

* 镖群：大概是一次发射多枚飞镖以增加命中率和伤害输出
* 君瞄：应该是组里某个大佬（可能是李君琪？）发明的自瞄算法
* 方瞄：同理，（可能是一位姓方的大佬）开发的自瞄或者其他算法
* 救命啊：可能是我方机器人正在遭到攻击，或者我方机器人被卡住了，需要其他车来抢救一下
* 整车预测：大概是对整个机器人的比赛表现进行预测，包括采取什么战术、大概有什么性能、可能遇到什么情况，从而优化比赛策略和操作（会不了一点）

梯度三：简答题 如果前两道题⽬你都能得⼼应⼿，那你有了解过这⼏个⽐较⼩众的词语是啥意思吗： ⽔⺟骑⼠ 8000轮腿哥

（一点都不得心应手……给点分给孩子把<doge>）

* 水母骑士：开发机器人算法的队员
* 8000轮腿哥：（？？？？？）这里直接下一个暴论，指赛场上特别能给队友安全感和信心的机器人或者操作手

第二题：

#include <iostream>

#include <vector>

#include <queue>

#include <cmath>

#include <iomanip>

using namespace std;

const int MAXN = 10;

const int INF = 1e9;

int grid[MAXN][MAXN];

double expandedGrid[MAXN][MAXN];

int dx[4] = { -1, 0, 1, 0 };

int dy[4] = { 0, 1, 0, -1 };

struct Node {

int x, y;

double cost;

bool operator>(const Node& other) const {

return cost > other.cost;

}//从而将Node的大小与cost绑定，方便后续比较和排序

};

void calculateExpandedGrid(int n, int m, double t, int x1, int y1) {

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

expandedGrid[i][j] = grid[i][j];

expandedGrid[x1][y1] = 0;

for (int k = 0; k < 4; ++k) {

int nx = i + dx[k];

int ny = j + dy[k];

if (nx >= 0 && nx < n && ny >= 0 && ny < m) {

expandedGrid[i][j] += t \* grid[nx][ny];

}

}

}

}

}

double dijkstra(int n, int m, int x1, int y1, int x2, int y2) {

priority\_queue<Node, vector<Node>, greater<Node>> pq;

vector<vector<double>> dist(n, vector<double>(m, INF));

vector<vector<bool>> visited(n, vector<bool>(m, false));

pq.push({ x1, y1, expandedGrid[x1][y1] });

dist[x1][y1] = expandedGrid[x1][y1];

while (!pq.empty()) {

Node current = pq.top();

pq.pop();

int cx = current.x;

int cy = current.y;

if (visited[cx][cy]) continue;

visited[cx][cy] = true;

if (cx == x2 && cy == y2) {

return dist[cx][cy];

}

for (int i = 0; i < 4; ++i) {

int nx = cx + dx[i];

int ny = cy + dy[i];

if (nx >= 0 && nx < n && ny >= 0 && ny < m && !visited[nx][ny]) {

double newCost = dist[cx][cy] + expandedGrid[nx][ny];

if (newCost < dist[nx][ny]) {

dist[nx][ny] = newCost;

pq.push({ nx, ny, newCost });

}

}

}

}

return -1;

}

int main() {

int n, m;

cin >> n >> m;

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

cin >> grid[i][j];

}

}

double t;

cin >> t;

int x1, y1, x2, y2;

cin >> x1 >> y1 >> x2 >> y2;

cout << endl;

calculateExpandedGrid(n, m, t,x1,y1);

for (int i = 0;i < 4;i++) {

for (int j = 0;j < 4;j++) {

cout << expandedGrid[i][j] << " ";

}

cout << endl;

}

double cost\_min = dijkstra(n, m, x1, y1, x2, y2);

cout << fixed << setprecision(2) << cost\_min << endl;

return 0;

}

第三题：

完全不会……

大致思路：  
1、获取当前位置和传感器探测到的距离（具体对于误差的处理可能是什么滤波？？？）

2、估计目标的运动模式

具体识别思路：（反正应该是用卡尔曼滤波，具体真没时间看完力）

静止——看上一帧和当前帧的位置是否在误差范围内重合

匀速直线——看位置差与时间是否成正比

三角函数——？？？？

3、根据估计的目标位置，利用SET\_TARGET\_POSITION函数设置机器人的目标位置

并且这个位置应该考虑延时（用上面检测到的模式进行公式推算）

具体实现——我不到啊

4、计算current\_error、average\_error和expotional\_error，评估跟随效果

并且if如果expotional\_error连续 5 秒大于 0.1，则根据目标模式进行重启操作

具体实现——？？？

第四题

见cpp

第五题：

见cpp