第四套试题

1.小型飞船(flyship.pas/c/cpp)

小白老师给大家留了道作业：有一批太空物资需要用小型飞船运送，每艘小型飞船每次最多只能运送总重量不超过w的两件物品。为了节约运费，一次性运送这批物资至少需多少艘小型飞船？聪明的你肯定能想出办法来哟！

【输入格式】

第一行共有2个正整数：w n (100 <= w <= 10 000)

第二行共有n个不超过w的正整数：依次表示n件物品的重量，相邻两数间用一个空格隔开。

【输出格式】

只有一行且只有一个正整数：最少的小型飞船数目。

第四套试题

1.小型飞船(flyship.pas/c/cpp)

【输入样例】

100 9

90 20 20 30 50 60 70 80 90

【输出样例】

6

【样例说明】

这6艘小型飞船的载重量可以是：90，90，80+20，70+30，60+20，50。

【数据范围】

50% 的数据: 1 <= n <= 100

80% 的数据: 1 <= n <= 100 000

100% 的数据: 1 <= n <= 1 000 000

第四套试题

1.小型飞船(flyship.pas/c/cpp)

思路:简单的贪心思想，将物品的重量排序（此处选用由小到大），如果最大重量+最小重量≤限制重量，那么这两个物品就能放在同一艘飞船上，不能则只能放最大的；再去查找剩下物品中最大+最小重量的物品能否放在同一艘飞船上。

```
sort(a+1,a+n+1);
int l=1,r=n,ans=0;
while(l<=r){
        if(a[r]+a[l]<=w) l++;
        r--;
        ans++;
}
```

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

Tao厌倦了不分昼夜的工作，决定玩一会宠物小精灵游戏来放松一下。这个游戏在某些地方也叫做精灵宝可梦

这款游戏系统会给你一些小精灵，它们有着自己的名字。每个小精灵都可以吃糖果来进化，每次进化完成后它会返还你2个糖果。

系统给了N个种类的小精灵，对于每个小精灵Pi，Tao准备了Mi个糖果给它进化用，Tao想知道这N个小精灵总共能进化几次。

除此之外，Tao还想知道哪个小精灵进化的次数最多。如果有多个进化次数相同的小精灵，输出最先出现的小精灵的编号.

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

【输入格式】

第1行：包含一个整数N（1≤N≤70）表示小精灵的种类数；

第2行至第2n+1行: 每两行为一组：

第一行包含一个字符串pi（1≤pi≤20），表示小精灵的种类名称；

第二行包含两个整数 Ki(1 ≤Ki≤ 400)和Mi(1 ≤ Mi≤ 104，Ki表示该种小精灵进化一次所需的糖果数，Mi表示Tao为该种小精灵准备的糖果数)；

【输出格式】

第一行包含一个整数，表示Tao所拥有的小精灵总共能进化多少次。

第二行包含一个字符串，表示哪种小精灵进化次数最多。

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

【输入样例】

```
4
Caterpie
12 33
Weedle
12 42
Pidgey
12 47
Rattata
25 71
```

【输出样例】

```
14
Weedle
```

【数据范围】

对于 100％的数据，（1≤N≤70，1 ≤Ki≤ 400，1 ≤ Mi≤ 104）。

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

思路：根据题意，可以看出本题采用模拟算法，如果某个小精灵所拥有的糖果数≥所需进化的糖果数，那么，它就进化一次，并返还2个糖果，如此反复，直到糖果不足以支持一次进化。在这N个小精灵进化完后，输出总进化次数和进化次数最多的小精灵名字。

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

程序实现：1.为了很好的表示小精灵名字、进化所需糖果数和拥有的糖果数，采用结构体变量存储；同时用数组来存储该小精灵进化次数。

```
struct Pokemon{
    string name;
    int candyReq;
    int candyNo;
};
Pokemon p[MAX_POKEMON];
int evolutionNo[MAX_POKEMON];
```

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

程序实现：2.对于某一个小精灵p[i]来说：

```
 while ( p[i].candyNo >= p[i].candyReq )
{
        p[i].candyNo -= p[i].candyReq;
        p[i].candyNo += 2;
        evolutionNo[i]++;
        totalEvolution++;
    }
```

第四套试题

2.精灵宝可梦（go.pas/c/cpp）

程序实现：3.比较该小精灵进化次数和最大进化次数,记录进化次数最多的小精灵编号。

```
maks=0;
 if ( evolutionNo[i] > maks ){
          maks=evolutionNo[i];
          maksi=i;
       }
```

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

北京地图可以看作是 R*C 的网格，奥运会期间对有的地方要进行交通管制，有的地方不允许进入，有的地方对离开时的行驶方向有限制：有的只允许走到上下两个相邻的格子，有的只允许走到左右两个相邻的格子，没有的任何限制的地方上下左右四个方向都允许。

现在给你地图的描述，格子的描述如下：

1.“+”表示可以向任意方向（上、下、左、右）移动一格；

2.“-”表示只能向左右方向移动一格；

3.“|”表示只能向上下方向移动一格；

4.“*”表示该位置不能到达。

你的任务是计算出从左上角到右下角的最少需要经过的格子数。

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

【输入格式】

包含t组测试数据：

第一行一个整数t(1<=t=10)，表示有t组测试数据。

每一个测试数据，第一行一个整数r，第二行一个整数 c，表示地图是 r 行 c 列；

接下来 r 行，每行 c 个字符，每个字符是{+,*,-,|}中的一种。

你可以假设左上角不会是“*”。

【输出格式】

输出有t行，每行一个整数表示对应测试数据所需的最少格子数，如果到达不了右下角输出-1。

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

【输入样例】

```
3
2
2
-|
*+
3
5
+||*+
+++|+
**--+
2
3
+*+
+*+
```

【输出样例】

```
3
7
-1
```

【数据范围】

对于50%的数据： 1<=r,c<=20

对于100%的数据： 1<=r,c<=1000

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

思路：题目大意：给定一个地图、起点和终点，求最短路径，很符合宽搜bfs的思想，数据范围1000*1000也满足时间复杂度限制，所以本题的算法为宽搜求最短路径。

本题在典型宽搜的基础上额外加了几个限制条件："|"和"—"；那么这两个限制条件只需类似"+"的拓展方式即可。

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

程序实现：

1.运用队列q[]实现宽搜的过程，那么宽搜的状态为地图的横纵坐标，那么q[].x为横坐标，q[].y为纵坐标。定义d[][]为到达某一点需走的格子数；定义vis[][]标记某一位置是否走过。

```
int R,C;
char g[1111][1111];
int vis[1111][1111],d[1111][1111];
```

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

程序实现：

2.状态拓展：分为三种情况：

“+”：上下左右四个方位入队；

“-”：左右方位入队；

“|”：上下方位入队；

以上拓展均需判断是否出地图、是否能走、是否走过。

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

程序实现：

```
if(g[r][c]=='+'){
        expand(r,c-1,dist);
        expand(r,c+1,dist);
        expand(r-1,c,dist);
        expand(r+1,c,dist);
}
else if(g[r][c]=='|'){
        expand(r-1,c,dist);
        expand(r+1,c,dist);
}
else if(g[r][c]=='-'){
        expand(r,c-1,dist);
        expand(r,c+1,dist);
}
```

```
int expand(int r,int c,int dist){
        if(r<0||c<0||r>=R||c>=C)
        return 0;
        if(vis[r][c]||g[r][c]=='*')
        return 0;
        vis[r][c]=1;
        d[r][c]=dist+1;
        q[tail++]=mkp(r,c);
        return 0;
}
```

第四套试题

3.北京地图 (bgmap.pas/c/cpp)

程序实现：

3.判断是否到达终点：只需判断vis[][]数组有没有标记终点即可。

```
if(!vis[R-1][C-1])
                d[R-1][C-1]=-1;
        printf("%d\n",d[R-1][C-1]);
```

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

国庆期间，小K在各个风景区游玩，并带回来了许多纪念品。她高兴坏了，但是她的爸爸并不允许家里放这么多“没用”的东西，经过两人激烈的争论，最终爸爸同意她有条件的放一些纪念品在家里。

条件是把所有纪念品按购买日期先后放在一排（即位置固定），每个纪念品都有一个价格，那么相邻两个纪念品的价格的差值（绝对值）不能为1（当然如果相邻两个纪念品的价格一样也没事），只有满足这个条件，爸爸才允许小K把这些纪念品放在家里。自然的，这意味着小K必须去掉一些纪念品才能满足爸爸给的条件，小K希望去掉的纪念品越少越好，但她又不知道怎么选，所以她请求你帮助她解决这个难题，让她能够留下最多的纪念品。

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

【输入格式】

第一行一个整数 N，表示纪念品数；

第 2~N+1 行 N 个整数，按照购买日期排下来，第 i 个为 ci。表示第 i 个 纪念品的价格。

【输出格式】

包含一个数，表示最少要去掉的纪念品数。

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

【输入样例】

6
4
2
2
1
1
1

【输出样例】

2

【数据范围】

对于 100％的数据，3 <= N <= 33

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

思路：本题类似于最长不下降子序列，找到最长的满足相邻两个差值不为1的序列，那么用总长度减去这个最长长度，得到的就是最终答案，舍弃最少的数目。

注：最长不下降子序列（LIS）题面为给定N个整数序列，找出最长的不下降子序列的长度。

LIS的状态转移方程为：

max{f[i],f[j]+1}(1<=j<=i-1&&l[j]<=l[i])

边界条件为：f[]=1;

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

思路：LIS的状态转移方程为：

max{f[i],f[j]+1}(1<=j<=i-1&&l[j]<=l[i])

边界条件为：f[]=1;

类比可以得到，本题的状态转移方程为：

max{f[i],f[j]+1}(1<=j<=i-1&&abs(l[j]-l[i])!=1)

边界条件为：f[]=1;

第四套试题

4.艰难取舍（seq.cpp/c/pas ）

程序实现：

```
for (int i=1;i<=n;i++)
        for (int j=1;j<=i-1;j++)
                if (abs(l[i]-l[j])!=1)
                        f[i]=max(f[j]+1,f[i]);//状态转移
        for (int i=1;i<=n;i++)
        {
                if (f[i]>maxn)  maxn=f[i];//找出留下来长度最长的
        }
        cout<<n-maxn<<endl; //用总长度减去最长的长度即为答案
```