CSP-S 初赛模拟题 （120 分钟）

1. 单项选择题（每题 2 分，15 题，共 30 分）

1. 若有定义 char x[]=“12345”;char y[]={‘1’,‘2’,‘3’,‘4’,‘5’};则（）。

   A. x 数组与 y 数组的所占内存空间相同

   B. x 数组比 y 数组占的内存空间大

   C. x 数组比 y 数组占的内存空间小

   D. x 数组等价于 y 数组

2. -128 的补码表示为( )

   A. 00000000　　B. 00000001　　C. 10000000　　D. 11111111

3. 给出 4 种排序:插入排序、冒泡排序、选择排序、快速排序。这 4 种排序的时间代价分别是(　　)。

   A. O(n^2)、O(n^2)、O(n^2)、O(nlogn)

   B. O(n^2)、O(n^2)、O(n^2)、O(logn)

   C. O(nlogn)、O(n^2)、O(n^2)、O(nlogn)

   D. O(nlogn) 、 O(n^2)、O(nlogn)、O(nlogn)

4. (2019)10+(2020)8 的结果是（）。

   A. (3049)10　　B. (BF3)16　　C. (101111110001)2　　D. (5765)8

5. 平面上有三条平行直线,每条直线上分别有 7,5,6 个点,且不同直线上三个点都不在同一条直线上。问用这些点为顶点,能组成(　　)个不同四边形。

   A. 18　　B. 210　　C. 2250　　D. 4500

6. 同时查找 2n 个数中的最大值和最小值，最少比较次数为()。

   A. 3(n-2)/2　　B. 3n-2　　C. 4n-2　　D. 2n-2

7. 将 2 个红球,1 个蓝球,1 个白球放到 10 个编号不同的盒子中去,每个盒子最多放一个球,有多少种放法( )。

   A. 5040　　B. 2520　　C. 1260　　D. 420

8. 一棵结点数为 2015 的二叉树最多有 _____ 个叶子结点。

   A. 1007　　B. 1008　　C. 1009　　D. 1006

9. G 是一个非连通简单无向图，共有 36 条边，则该图至少有()个顶点

   A. 10　　B. 9　　C. 8　　D. 7

10. 由四个不同的点构成的简单无向连通图的个数是( )

    A. 32　　B. 35　　C. 38　　D. 31

11. 前缀表达式- + * 4 + 2 3 1 5 的值为()

    A. 16　　B. 17　　C. 19　　D. 15

12. 2+3*(4-(5+6))/7 的逆波兰表达式为( )

    A 2 3 4 5 6 - + * 7 / +

    B. 2 3 4 5 6 - + * / 7 +

    C. 2 3 4 5 6 + - * 7 / +

    D. 2 3 4 5 6 + + * / 7 -

13. 已知 T(n)=T (n/2)+1，T(1)=1 求时间复杂度（）

A.O（n）　B.O（logn）　C.O（nlogn）　D.O（n^2logn）

14. 公共汽车起点站于每小时的 10 分，30 分，55 分发车，该顾客不知发车时间，在每小时内的任一时刻随机到达车站，则乘客候车时间的数学期望（精确到秒）是（）。

A.8 分 40 秒　B.15 分 20 秒　C.22 分 30 秒　D.10 分 25 秒

15. 设要将序列<Q, H, C, Y, P, A, M, S, R, D, F, X>中的关键码按字母的升序重新排列，则（　　）是以第一个元素为分界元素的快速排序一趟扫描的结果。

A.F, H, C, D, P, A, M, Q, R, S, Y, X

B.P, A, C, S, Q, D, F, X, R, H, M, Y

C.A, D, C, R, F, Q, M, S, Y, P, H, X

D.H, C, Q, P, A, M, S, R, D, F, X, Y

2. 阅读程序题（共 40 分）

1. 阅读程序题 1，请阅读程序，回答问题。

```
#include <cstdio>
using namespace std;
int findvall(int n)
{
    int f;
    if (n == 0) return 1;
    else
    {
        f = findvall(n / 2);
        return (n*f);
    }
}
int main()
{
    int n;
    scanf("%d", &n);
    printf("%d\n", findvall(n));
    return 0;
}
```

1.第 6 行输出 if(n==0)改成 if(n==1)时对于输入的正整数 n,输出结果不会改变。()

A.正确　B.错误

2.对于输入的正整数程序输出的值小于等于 n。()

A.正确　B.错误

3.如果输入的 n 是负数的话,该程序会出现死循环,所以该程序不能求解 n 是负数的情况。()

A.正确　B.错误

4.如果多次运行该程序,并且输入的 n 是单调递增的正整数,那么每次输出的结果也是一个严格单调递增的数列。()

A.正确　B.错误

5. 若两次输入 n 的值相差 1,但输出的结果却是 1 个正数,一个负数,那么两次输入的 n 可能是下面四组中的()。

A.不可能　B.-6,-7　C.-15,-16　D.-23,-24

6.此程序的时间复杂度是（）。

A.O(n^2)　　B.O(logn)　　C.O(n)　　D.O(nlogn)

```
2.
#include <iostream>
#include <cmath>
#define MAX 1000
#define p sqrt(3)
using namespace std;
int n,dp[1000][3];
int h0=1,h1=3;
double ans1=(2+p)/(2*p),ans2=(-2+p)/(2*p);
int main(){
    cin>>n;
    dp[1][0]=dp[1][1]=dp[1][2]=1;
    for(int i=2,tmp;i<=n;i++)
    {
        dp[i][0]=dp[i-1][1]+dp[i-1][2];
        dp[i][1]=dp[i-1][0]+dp[i-1][1]+dp[i-1][2];
        dp[i][2]=dp[i-1][0]+dp[i-1][1]+dp[i-1][2];
        tmp=h1;
        h1=2*(h1+h0);
        h0=tmp;
    }
    for(int i=1;i<=n;i++){
        ans1=ans1*(1+p);
        ans2=ans2*(1-p);
    }
    cout<<h1<<endl;
    cout<<dp[n][0]+dp[n][1]+dp[n][2]<<endl;
    cout<<ans1+ans2<<endl;
    return 0;
}
```

2.1　上述程序的输出中 h1 和 dp[n][0]+dp[n][1]+dp[n][2]的值相等

对

错

2.2　上述程序的输出中 dp[n][0]+dp[n][1]+dp[n][2]和 ans1+ans2 的值相等

对

错

2.3　当 n 等于 5 时，第一行输出(即 h1)结果为( )

A 164　　B.60　　C.448　　D.128

2.4 当 n 等于 10 时，第三行输出(即 ans1+ans2)结果为( )

A.9136　　B.68192　　C.24960　　D.3344

3.阅读程序题 3，请阅读程序，回答问题。

本题是一款模拟贪吃蛇程序,游戏是在一个 a*a 的网格上进行的。其中输入第一行一个整数 a。第二行两个整数 n 和 m。接下来是 n 行,每行第一个数为 opt,表示操作编号。接下来的输入的变量与操作编号对应,输出:即第 m 秒过后的地图,蛇所在的位置输出“o”,其余位置输出".", 以换行结尾。

```
#include <bits/stdc++.h>
#include <windows.h>
using namespace std;
int a, mp[101][101];
int t[100003];
int y[100003];
```

```
int cnt;
int len = 2, dir = 3, die = 0;
const int dx[5] = {0, 0, -1, 0, 1};
const int dy[5] = {0, -1, 0, 1, 0};
int nx = 0, ny = 1;
int px = 1, py = 2;
int check(int x, int yy)
{
    if (x < 1 || x > a || yy < 1 || yy > a)
        return 1;
    if (cnt + 1 - mp[x][yy] < len)
        return 1;
    return 0;
}
void work()
{
    if (die)
        return;
    px += nx;
    py += ny;
    die = check(px, py);
    if (die)
        return;
    mp[px][py] = ++cnt;
}
void show()
{
    for (int i = 1; i <= a; ++i)
    {
        for (int j = 1; j <= a; ++j)
        {
            if (mp[i][j] != 0 && mp[i][j] >= cnt - len + 1)
                putchar('o');
            else
                putchar('.');
        }
        puts("");
    }
}
int main()
{
    mp[1][1] = ++cnt;
    mp[1][2] = ++cnt;
    int n, m, op, xx;
    char s[3];
    scanf("%d", &a);
    scanf("%d%d", &n, &m);
    while (n--)
    {
        scanf("%d%d", &op, &xx);
        if (op == 1)
        {
            t[xx] = 1;
            scanf("%s", s);
            if (s[0] == 'L')
                y[xx] = 1;
            else if (s[0] == 'U')
                y[xx] = 2;
            else if (s[0] == 'R')
                y[xx] = 3;
```

```
            else
                y[xx] = 4;
        }
        else
        {
            t[xx] = 2;
        }
    }
    for (int tm = 1; tm <= m; ++tm)
    {
        if (t[tm] == 1)
        {
            if (y[tm] % 2 != dir % 2)
            {
                dir = y[tm];
                nx = dx[y[tm]];
                ny = dy[y[tm]];
            }
        }
        else if (t[tm] == 2)
        {
            ++len;
        }
        work();
        if (die)
        {
            break;
        }
    }
    show();
    return 0;
}
```

1.由程序代码可知,贪吃蛇的初始长度为 2,蛇头和蛇尾分别在坐标[1,2]、[1,1]处 ( )。

A.正确　　B.错误

2.check 函数是用来检测蛇是否吃到果实的。()

A.正确　　B.错误

3.第 54 行及第 58 行输入 1 x y 表示在第 x 秒按下了 y 键,y 为 LURD 中的一种,分别表示按下了左、上、右、下四种按钮()。

A.正确　　B.错误

4. 当输入样例如下所示时:

```
10
10 20
2 1
2 2
2 3
2 4
2 5
1 6 R
1 7 D
1 8 L
1 9 U
```

最终程序的运行结果所代表的含义可表示为贪吃蛇在第 9 秒过后就死亡了,因此最后贪吃蛇保持的是死亡前(第 7 秒过后)的位置()。

A.正确　　B.错误

5. 若输入地图边长为 x,共 n 次操作(x>n),该程序时间复杂度为()。

A.x^2　　B.n^2　　C.n^2* x　　D.x^2* n

3. 完善程序（共 30 分）

1. 过河问题

在一个月黑风高的夜晚，有一群人在河的右岸，想通过唯一的一根独木桥走到河的左岸．在伸手不见五指的黑夜里，过桥时必须借照灯光来照明，不幸的是，他们只有一盏灯．另外，独木桥上最多能承受两个人同时经过，否则将会坍塌．每个人单独过独木桥都需要一定的时间，不同的人要的时间可能不同．两个人一起过独木桥时，由于只有一盏灯，所以需要的时间是较慢的那个人单独过桥所花费的时间．现在输入 N(2<=N<1000) 和这 N 个人单独过桥需要的时间，请计算总共最少需要多少时间，他们才能全部到达河左岸．

例如，有 3 个人甲、乙、丙，他们单独过桥的时间分别为 1、2、4，则总共最少需要的时间为 7．具体方法是：甲、乙一起过桥到河的左岸，甲单独回到河的右岸将灯带回，然后甲、丙在一起过桥到河的左岸，总时间为 2+1+4=7．

```
#include <iostream>
#include<cstring>
using namespace std;
const int SIZE=100;
const int INFINITY = 10000;
const bool LEFT=true;
const bool RIGHT =false;
const bool LEFT_TO_RIGHT=true;
const bool RIGHT_TO_LEFT=false;
int n,hour[SIZE];
bool pos[SIZE];
int max(int a,int b)
{
    if(a>b)
       return a;
    else
       return b;
}
int go(bool stage)
{
    int i,j,num,tmp,ans;
    if(stage==RIGHT_TO_LEFT)
    {
        num=0;
        ans=0;
        for(i=1;i<=n;i++)
           if(pos[i]==RIGHT)
           {
               num++;
               if( hour[i]>ans)
                  ans=hour[i];
           }
        if( ______①______ )
            return ans;
        ans=INFINITY;
        for(i=1;i<=n-1;i++)
           if(pos[i]==RIGHT)
               for(j=i+1;j<=n;j++)
                  if(pos[j]==RIGHT)
                  {
                      pos[i]=LEFT;
                      pos[j]=LEFT;
                      tmp=max(hour[i],hour[j])+ ______②______;
```

```
                    if(tmp<ans)
                       ans=tmp;
                    pos[i]=RIGHT;
                    pos[j]=RIGHT;
                }
        return ans;
    }
    if(stage==LEFT_TO_RIGHT)
    {
        ans=INFINITY;
        for(i=1;i<=n;i++)
            if( ______③______ )
            {
                pos[i]=RIGHT;
                tmp=_____④_____;
                if(tmp<ans)
                    ans=tmp;
                _____⑤_____;
            }
        return ans;
    }
    return 0;
}
int main()
{
    int i;
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cin>>hour[i];
        pos[i]=RIGHT;
    }
    cout<<go[RIGHT_TO_LEFT)<<endl;
    return 0;
}
```

1）①处应填（）

A. num < 2　　B. num <= 2

C. num >= 2　　D. num > 2

2）②处应填（）

A. go(RIGHT)　　B. go(RIGHT_TO_LEFT)

C. go(LEFT_TO_RIGHT)　　D. go(LEFT)

3）③处应填（）

A. pos[i] == LEFT_TO_RIGHT　　B. pos[i] == RIGHT_TO_LEFT

C. pos[i] == RIGHT　　D. pos[i] == LEFT

4）④处应填（）

A. hour[i] + go(RIGHT_TO_LEFT)　　B. hour[i] + go(LEFT_TO_RIGHT)

C. go(RIGHT_TO_LEFT)　　D. go(LEFT_TO_RIGHT)

5）⑤处应填（）

A. pos[i] = RIGHT　　B. pos[i] = LEFT

C. pos[i] = ++RIGHT　　D. pos[i] = ++LEFT

2.烽火传递

烽火台又称烽燧，是重要的军事防御设施，一般建在险要处或交通要道上。一旦有敌情发生，白天燃烧柴草，通过浓烟表达信息；夜晚燃烧干柴，以火光传递军情。在某两座城市之间有 n 个烽火台，每个烽火台发出信号都有一定的代价。为了使情报准确地传递，在连续的 m 个烽火台中至少要有一个发出信号。现输入 n、m 和每个烽火台发出信号的代价，请计算总共最少花费多少代价，才能使敌军来袭之时，情报能在这两座城市之间准确传递。

例如，有 5 个烽火台，他们发出信号的代价依次为 1，2，5，6，2，且 m 为 3，则总共最少花费代价为 4，即由第 2 个和第 5 个烽火台发出信号。

```
#include <iostream>
#include<cstring>
using namespace std;
const int SIZE=100;
int n,m,r,value[SIZE],heap[SIZE],
    pos[SIZE],home[SIZE],opt[SIZE];
    //heap[i]表示用顺序数组储存的堆 heap 中第 i 个元素的值
    //pos[i]表示 opt[i]在堆 heap 中的位置，即 heap[pos[i]]=opt[i]
    //home[i]表示 heap[i]在序列 opt 中的位置，即 opt[home[i]]=heap[i]
void swap(int i,int j)//交换堆中的第 i 个和第 j 个元素
{
    int tmp;
    pos[home[i]]=j;
    pos[home[j]]=i;
    tmp=heap[i];
    heap[i]=heap[j];
    heap[j]=tmp;
    tmp=home[i];
    home[i]=home[j];
    home[j]=tmp;
}
void add(int k)//在堆中插入 opt[k]
{
    int i;
    r++;
    heap[r]= _____①_____;
    pos[k]=r;
    _____②_____;
    i=r;
    while( (i>1) && (heap[i]<heap[i/2]) )
    {
        swap(i,i/2);
        i/=2;
    }
}
void remove(int k)//在堆中删除 opt[k]
{
    int i,j;
    i=pos[k];
    swap(i,r);;
    r--;
    if(i==r+1)
      return ;
    while( (i>1)&&(heap[i]<heap[i/2]) )
    {
        swap(i,i/2);
        i/=2;
    }
```

```
    while(i+i<=r)
    {
        if( (i+i+1<=r) && (heap[i+i+1]<heap[i+i]) )
            j=i+i+1;
        else
            _____③_____;
        if(heap[i]>heap[j])
        {
            _____④_____;
            i=j;
        }
        else
            break;
    }
}
int main()
{
    int i;
    cin>>n>>m;
    for(i=1;i<=n;i++)
        cin>>value[i];
    r=0;
    for(i=1;i<=m;i++)
    {
        opt[i]=value[i];
        add(i);
    }
    for(i=m+1;i<=n;i++)
    {
        opt[i]= _____⑤_____;
        remove(_____⑥_____);
        add(i);
    }
    cout<<heap[1]<<endl;
    return 0;
}
```

1）①处应填（）

A. opt[k]　　B. k　　C. heap[k]　　D. value[k]

2）②处应填（）

A. home[k] = r　　B. home[r] = k

C. home[r] = opt[k]　　D. home[k] = opt[r]

3）③处应填（）

A. j = i + i+1　　B. j = i + i　　C. j = i + i-1　　D. j = i +1

4）④处应填（）

A. swap(heap[i],heap[j])　　B. swap(heap[j],heap[i])

C. swap(i, j)　　D.

5）⑤处应填（）

A. value[i]　　B. heap[1]　　C. value[i] + heap[1]　　D. value[i] - heap[1]

6）⑥处应填（）

A. i+m　　B. i-m　　C. i　　D. m

3.尺取法求区间个数

给 n 个非负整数 a[1]，a[2]，…，a[n]，求区间和小于或等于 k 的区间个数，

即求使 SUM=a[L]+a[L+1]+…+a[R-1]+a[R]<=k 的区间[L,R]的个数(1<=L<=R<=n)，但由于对内存和复杂度有要求，本题已经用尺取法写好部分代码,请补全程序。

输入：

第一行两个整数 n，k(1<=n<=1000000，0<=k<=10000000000000000)。

第二行为 n 个数，表示 a[1]~a[n]的值(0<=a[i]<=1000000000)。

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int mx = 1e6 + 10;
int n, a[mx];
ll k, sum, ans;
int main()
{
    scanf("%d%lld", &n, &k);
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &a[i]);
    }
    int r = 0;
    for (int i = 1; i <= n; ++i)
    {
        while (r < n)
        {
            if (①)  ②;
            else break;
        }
        ③;
        if (i <= r)  ④;
        else  ⑤;
    }
    printf("%lld\n", ans);
}
```

1.①处应填（）。

A.sum+a[r+1]<=k　　B.sum+a[r]<=k

C.sum+a[r+1]<k　　D.sum+a[r]<k

2. ②处应填（）。

A.sum+=a[r]　　B.sum+=a[++r]

C.sum+=a[r++]　　D.sum+=a[r+1]

3.③处应填（）。

A.ans+=r-i+1　　B.ans+=r-i

C.ans+=r-i-1　　D.ans+=n-i+1

4.④处应填（）。

A.sum+=a[i]　　B.sum=a[r]

C.sum=a[i]　　D.sum-=a[i]

5.⑤处应填（）。

A.r = ++i;　　B.r = i--;

C.r = i++;　　D.r = i;