

第十六届全国青少年信息学奥林匹克联赛初赛试题

(提高组 C++语言 两小时完成)

●● 全部试题答案均要求写在答卷纸上，写在试卷纸上一律无效 ●●

一、单项选择题 (共 10 题，每题 1.5 分，共计 15 分。每题有且仅有一个正确选项。)

1. 与十六进制数 A1.2 等值的十进制数是 ()。
A. 101.2 B. 111.4 C. 161.125 D. 177.25
2. 一个字节 (byte) 由 () 个二进制组成。
A. 8 B. 16 C. 32 D. 以上都有可能
3. 以下逻辑表达式的值恒为真的是 ()。
A. $P \vee (\neg P \wedge Q) \vee (\neg P \wedge \neg Q)$ B. $Q \vee (\neg P \wedge Q) \vee (P \wedge \neg Q)$
C. $P \vee Q \vee (P \wedge \neg Q) \vee (\neg P \wedge Q)$ D. $P \vee \neg Q \vee (P \wedge \neg Q) \vee (\neg P \wedge \neg Q)$
4. Linux 下可执行文件的默认扩展名是 ()。
A. exe B. com C. dll D. 以上都不是
5. 如果在某个进制下等式 $7*7=41$ 成立，那么在该进制下等式 $12*12=$ () 也成立。
A. 100 B. 144 C. 164 D. 196
6. 提出“存储程序”的计算机工作原理的是 ()。
A. 克劳德·香农 B. 戈登·摩尔 C. 查尔斯·巴比奇 D. 冯·诺依曼
7. 前缀表达式 “+ 3 * 2 + 5 12” 的值是 ()。
A. 23 B. 25 C. 37 D. 65
8. 主存储器的存取速度比中央处理器 (CPU) 的工作速度慢的多，从而使得后者的效率受到影响。而根据局部性原理，CPU 所访问的存储单元通常都趋于一个较小的连续区域中。于是，为了提高系统整体的执行效率，在 CPU 中引入了 ()。
A. 寄存器 B. 高速缓存 C. 闪存 D. 外存
9. 完全二叉树的顺序存储方案，是指将完全二叉树的结点从上到下、从左到右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置上，则第 k 号结点的父结点如果存在的话，应当存放在数组中的 () 号位置。
A. 2k B. 2k+1 C. k/2 下取整 D. (k+1)/2

10. 以下竞赛活动中历史最悠久的是()。

- A. 全国青少年信息学奥林匹克联赛(NOIP)
- B. 全国青少年信息学奥林匹克竞赛(NOI)
- C. 国际信息学奥林匹克竞赛(IOI)
- D. 亚太地区信息学奥林匹克竞赛(APIO)

二、不定项选择题 (共 10 题, 每题 1.5 分, 共计 15 分。每题正确答案的个数不少于 1。多选或少选均不得分)。

1. 元素 R1、R2、R3、R4、R5 入栈的顺序为 R1、R2、R3、R4、R5。如果第 1 个出栈的是 R3, 那么第 5 个出栈的可能是()。

- A. R1
- B. R2
- C. R4
- D. R5

2. Pascal 语言, C 语言和 C++语言都属于()。

- A. 高级语言
- B. 自然语言
- C. 解释性语言
- D. 编译性语言

3. 原地排序是指在排序过程中(除了存储待排序元素以外的)辅助空间的大小与数据规模无关的排序算法。以下属于原地排序的有()。

- A. 冒泡排序
- B. 插入排序
- C. 基数排序
- D. 选择排序

4. 在整数的补码表示法中, 以下说法正确的是()。

- A. 只有负整数的编码最高位为 1
- B. 在编码的位数确定后, 所能表示的最小整数和最大整数的绝对值相同
- C. 整数 0 只有一个唯一的编码
- D. 两个用补码表示的数相加时, 如果在最高位产生进位, 则表示运算溢出

5. 一颗二叉树的前序遍历序列是 ABCDEFG, 后序遍历序列是 CBFEGDA, 则根结点的左子树的结点个数可能是()。

- A. 0
- B. 2
- C. 4
- D. 6

6. 在下列 HTML 语句中, 可以正确产生一个指向 NOI 官方网站的超链接的是()。

- A. `欢迎访问 NOI 网站`
- B. `欢迎访问 NOI 网站`
- C. `<a>http://www.noi.cn`
- D. `欢迎访问 NOI 网站`

7. 关于拓扑排序, 下列说法正确的是()。

- A. 所有连通的有向图都可以实现拓扑排序
- B. 对同一个图而言, 拓扑排序的结构是唯一的
- C. 拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面
- D. 拓扑排序结果序列中的第一个结点一定是入度大于 0 的点

8. 一个平面的法线是指与该平面垂直的直线。过点 (1, 1, 1)、(0, 3, 0)、(2, 0, 0) 的平面的法线是()。

- A. 过点 (1, 1, 1)、(2, 3, 3) 的直线
- B. 过点 (1, 1, 1)、(3, 2, 1) 的直线
- C. 过点 (0, 3, 0)、(-3, 1, 1) 的直线
- D. 过点 (2, 0, 0)、(5, 2, 1) 的直线

9. 双向链表中有两个指针域 llink 和 rlink, 分别指向该结点的前驱及后继。设 p 指向链表中的一个结点, 他的左右结点均为非空。现要求删除结点 p, 则下列语句序列中正确的是()。

- A. p->rlink->llink=p->rlink;
p->llink->rlink=p->llink; delete p;
- B. p->llink->rlink=p->rlink;
p->rlink->llink = p->llink; delete p;
- C. p->rlink->llink = p->llink;
p->rlink->llink->rlink = p->rlink; delete p;
- D. p->llink->rlink = p->rlink;
p->llink->rlink->link = p->llink; delete p;

10. 今年(2010 年)发生的事件有()。

- A. 惠普实验室研究员 Vinay Deolalikar 自称证明了 $P \neq NP$
- B. 英特尔公司收购计算机安全软件公司迈克菲(McAfee)
- C. 苹果公司发布 iPhone 4 手机
- D. 微软公司发布 Windows 7 操作系统

三. 问题求解 (共 2 题, 每空 5 分, 共计 10 分)

1. LZW 编码是一种自适应词典编码。在编码的过程中, 开始时只有一部基础构造元素的编码词典, 如果在编码的过程中遇到一个新的词条, 则该词条及一个新的编码会被追加到词典中, 并用于后继信息的编码。

举例说明, 考虑一个待编码的信息串: “xyx yy yy xyx”。初始词典只有 3 个条目, 第一个为 x, 编码为 1; 第二个为 y, 编码为 2; 第三个为空格, 编码为 3; 于是串 “xyx” 的编码为 1-2-1 (其中-为编码分隔符), 加上后面的一个空格就是 1-2-1-3。但由于有了一个空格, 我们就知道前面的 “xyx” 是一个单词, 而由于该单词没有在词典中, 我们就可以自适应的把这个词条添加到词典里, 编码为 4, 然后按照新的词典对后继信息进行编码, 以此类推。于是, 最后得到编码: 1-2-1-3-2-2-3-5-3-4。

我们可以看到, 信息被压缩了。压缩好的信息传递到接受方, 接收方也只要根据基础词典就可以完成对该序列的完全恢复。解码过程是编码过程的逆操作。现在已知初始词典的 3 个条目如上述, 接收端收到的编码信息为 2-2-1-2-3-1-1-3-4-3-1-2-1-3-5-3-6, 则解码后的信息串是 “_____”。

2. 无向图 G 有 7 个顶点, 若不存在由奇数条边构成的简单回路, 则它至多有_____条边。

3. 记 T 为一队列, 初始时空, 现有 n 个总和不超过 32 的正整数依次入列。如果无论这些数具体为何值, 都能找到一种出队的方式, 使得存在某个时刻队列 T 中的数之和恰好为 9, 那么 n 的最小值是_____。

四. 阅读程序写结果（共 4 题，每题 8 分，共计 32 分）

1.

```
#include<iostream>
using namespace std;
int main()
{
    const int SIZE=10;
    int data[SIZE],i,j,cnt,n,m;
    cin>>n>>m;
    for(i=1;i<=n;i++)
        cin>>data[i];
    for(i=1;i<=n;i++)
    {
        cnt=0;
        for(j=1;j<=n;j++)
            if( (data[i]<data[j]) || (data[j]==data[i] && j<i) )
                cnt++;
        if (cnt==m)
            cout<<data[i]<<endl;
    }
    return 0;
}
```

输入:

5 2

96 -8 0 16 87

输出: _____

2.

```
#include<iostream>
using namespace std;
int main()
{
    const int SIZE=100;
    int na,nb,a[SIZE],b[SIZE],i,j,k;
    cin>>na;
    for(i=1;i<=na;i++)
        cin>>a[i];
    cin>>nb;
    for(i=1;i<=nb;i++)
        cin>>b[i];
    i=1;
    j=1;
```

```

while( (i<=na)&&(j<=nb) )
{
    if(a[i]<=b[j])
    {
        cout<<a[i]<<' ';
        i++;
    }
    else
    {
        cout<<b[j]<<' ';
        j++;
    }
}
if(i<=na)
    for(k=i;k<=na;k++)
        cout<<a[k]<<' ';
if(j<=nb)
    for(k=j;k<=nb;k++)
        cout<<b[k]<<' ';
return 0;
}

```

输入:

5

1 3 5 7 9

4

2 6 10 14

输出: _____

3.

```
#include<iostream>
```

```
using namespace std;
```

```
const int NUM=5;
```

```
int r(int n)
```

```
{
```

```
    int i;
```

```
    if(n<=NUM)
```

```
        return 0;
```

```
    for(i=1;i<=NUM;i++)
```

```
        if( r(n-i)<0)
```

```
            return i;
```

```
    return -1;
```

```
}
```

```
int main()
{
    int n;
    cin>>n;
    cout<<r(n)<<endl;
    return 0;
}
```

输入:

1 6

输出: _____

4.

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE=100;
int n,m,r[SIZE];
bool map[SIZE][SIZE],found;
bool successful()
{
    int i;
    for(i=1;i<=n;i++)
        if(!map[r[i]][r[i%n+1]])
            return false;
    return true;
}
void swap(int *a,int *b)
{
    int t;
    t=*a;
    *a=*b;
    *b=t;
}
void perm(int left,int right)
{
    int i;
    if(found)
        return ;
    if(left>right)
    {
        if(successful())
        {
            for(i=1;i<=n;i++)
                cout<<r[i]<<' ';
```

```

        found=true;
    }
    return ;
}
for(i=left;i<=right;i++)
{
    swap(r+left,r+i);
    perm(left+1,right);
    swap(r+left,r+i);
}
}
int main()
{
    int x,y,i;
    cin>>n>>m;
    memset(map,false,sizeof(map));
    for(i=1;i<=m;i++)
    {
        cin>>x>>y;
        map[x][y]=true;
        map[y][x]=true;
    }
    for(i=1;i<=n;i++)
        r[i]=i;
    found=false;
    perm(1,n);
    if(!found)
        cout<<"No solution!"<<endl;
    return 0;
}

```

输入:

```

9 12
1 2
2 3
3 4
4 5
5 6
6 1
1 7
2 7
3 8
4 8
5 9

```

输出: _____

五. 完善程序 (第 1 题, 每空 2 分, 第 2 题, 每空 3 分, 共 28 分)

1. (过河问题) 在一个黑风高的夜晚, 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 ($2 \leq N < 1000$) 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;
}

```

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];
//hep[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];
    head[i]=head[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;
}

```