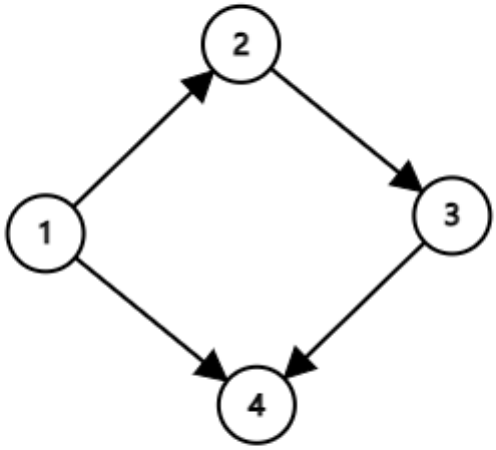
普及组初赛模拟题（一）

一、单项选择题（共15题，每题2分，共计30分；每题有且仅有一个正确选项）

1. 现代计算机所应用的储存程序原理是由谁提出的：
   - A. 图灵
   - B. 冯·诺依曼
   - C. 弗洛伊德
   - D. 高斯
2. 在单链表指针为 p 的结点之后插入指针为 s 的结点，正确的操作是:
   - A. `p->next=s;s->next=p->next;`
   - B. `s->next=p->next;p->next=s;`
   - C. `p->next=s;p->next=s->next;`
   - D. `p->next=s->next;p->next=s;`
3. 下列无符号数中，最小的数是：
   - A. $(1111001)_2$
   - B. $(171)_8$
   - C. $(119)_{10}$
   - D. $(79)_{16}$
4. 字符 `A` 的 ASCII 码为 65，则字符 `a` 的 ASCII 码为：
   - A. 95
   - B. 96
   - C. 97
   - D. 98
5. 设 $a[1]=4, a[2]=5, a[3]=7, a[4]=2$，那么表达式 $a[a[4]]+a[a[1]]$ 的值为：
   - A. 6
   - B. 7
   - C. 8
   - D. 9
6. 如图所示，如果从结点 1 开始进行广度优先搜索，那么最后被搜索到的结点是：



- A. 1
- B. 2
- C. 3
- D. 4

7. 一棵 n 层的满二叉树的节点数目：
   - A. $2^n - 1$
   - B. $2n - 1$
   - C. 2^{n-1}
   - D. $2n$
8. 两个长度分别为 n, m 的大整数相乘，得到的结果最长的长度可能为：
   - A. $n + m$
   - B. $n + m - 1$
   - C. $n + m + 1$
   - D. $n \times m$
9. 从 3 位老师和 9 位学生中选出 2 位老师，4 位同学去参加集训，共有多少种选择方式：
   - A. 378
   - B. 379
   - C. 648
   - D. 328

10. 希望幼儿园和光明幼儿园有分别有 243 个和 256 个同学，现在你要准备若干个蛋糕，为了公平，分给希望幼儿园和光明幼儿园的蛋糕数目要一样多，而且分给每个相同幼儿园的小朋友的蛋糕数目也要一样多，你至少要准备多少蛋糕：
    - A. 124412
    - B. 62206
    - C. 62208
    - D. 124416
11. 法国数学家爱德华·卢卡斯于 1883 年发明了一个叫河内塔的智力游戏：给定一个由 8 个圆盘组成的塔，这些圆盘按照大小递减的方式套在三根桩柱中的一根上，我们的目的是要将整个塔移动到另一根桩柱上，每次只能移动一个圆盘，且较大的圆盘在移动过程中不能放置在较小的圆盘上面，则最少需要移动的次数为：
    - A. 255
    - B. 63
    - C. 127
    - D. 511
12. 如果一个数不能被任何小于它且大于 2 的数整除，那我我们称这个数是奇异的数，下列是奇异的数的是：
    - A. 18
    - B. 124
    - C. 97943
    - D. 293829
13. 冒泡排序是一种经典的排序算法，其主要思想是顺序比较相邻的两位，这样我们在第 1 轮的排序结束之后，序列的最后一位一定会是最大的数，这样我们如果将一个长度为 n 的序列进行冒泡排序，那么我们为保证算法的正确性，我们最少需要进行多少轮的比较：
    - A. n
    - B. $n+1$
    - C. $n \times (n-1)$
    - D. $n-1$
14. 斐波那契数列是一个经典的问题，其递推公式为 $f_i = f_{i-1} + f_{i-2}$，现在我们知道 $f_1 = f_2 = 1$，那么 f_8 为：
    - A. 8
    - B. 5
    - C. 34
    - D. 21
15. 队列作为一种基本的数据结构，在很多场景下有着重要的应用，对于队列元素的进出方式，下列说法正确的是：
    - A. 先进后出
    - B. 先进先出

- C. 右进左出
- D. 左进右出

二、阅读程序（程序输入不超过数组或字符串定义的 范围；判断题正确填 T，错误填 F；除特殊说明外， 判断题 1.5 分，选择题 3 分，共计 40 分）

注意：判断题正确填 `T` ，错误填 `F` 。

阅读下面程序，完成第 $16 \sim 21$ 题。

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

using namespace std;

const int N = 1e3 + 5;

int n, m;
int c[N][N];

int main() {
    cin >> n >> m;
    for (int i = 0; i <= n; ++i) c[i][0] = 1;
    for (int i = 1; i <= n; ++i) { // (1)
        for (int j = 1; j <= n; ++j) { // (2)
            c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
        }
    }
    cout << c[n][m] << endl;
    return 0;
}

```

16. 若输入 `5 3` ，则输出 `10` 。[ ]
17. 若交换 `(1)` 和 `(2)` 所在行，不影响代码实现正确性。[ ]
18. 若程序输入的 n 小于 m ，则输出一定为 0。[ ]
19. 若输入 `0 0` ，则输出 `0` 。[ ]
20. 当满足 n 大于 m 时，若将该程序中的 `cout << c[n][m] << endl;` 修改为 `cout << c[n][n-m] << endl;` ，则一定不会改变程序的运行结果。[ ]
21. 以下问题可用上述算法进行求解的是 [ ]。
    - A. 将 n 个数的集合划分为 m 个非空子集的方案数
    - B. 从 n 个数中选 m 个数的方案数
    - C. 将 n 个数的集合划分为 m 个轮换的方案数（一个轮换就是一个首尾相接的环形排列）

- D. 以上均正确

阅读下面程序，完成第 $22 \sim 27$ 题。

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

using namespace std;

const int N = 1e5 + 5;

int f[N];

int main() {
    int n;
    cin >> n;
    f[0] = 1;
    for (int i = 1; i <= n; ++i) {
        for (int j = i; j <= n; ++j) {
            f[j] += f[j - i];
        }
    }
    cout << (f[n] - 1) << endl;
    return 0;
}
```

22. 若输入 `7` ，则输出 `14` 。[ ]
23. $f[i]$ 表示自然数 i 的加法分解（分解出来的整数个数至少为 2）方案数。[ ]
24. 如果输入的 n 较大，则该数组 f 会因值过大而产生负值。[ ]
25. 该程序的时间复杂度为[ ]。
    - A. $\mathcal{O}(n)$
    - B. $\mathcal{O}(n^2)$
    - C. $\mathcal{O}(nlogn)$
    - D. $\mathcal{O}(logn)$
26. 当输入为 `0` 时，输出为[ ]。
    - A. -1
    - B. 0
    - C. 1
    - D. 2
27. （4分）如果答案要对一个正整数 MOD 取余，那么下列写法最稳妥的是[ ]。
    - A. `f[j]+=f[j-i]%MOD; cout<<(f[n]-1)%MOD<<endl;`
    - B. `f[j]+=f[j-i]%MOD; cout<<(f[n]-1+MOD)%MOD<<endl;`
    - C. `(f[j]+=f[j-i])%=MOD; cout<<(f[n]-1)%MOD<<endl;`
    - D. `(f[j]+=f[j-i])%=MOD; cout<<(f[n]-1+MOD)%MOD<<endl;`

阅读下面程序，完成第 $28 \sim 33$ 题。

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

using namespace std;

const int N = 1e5 + 5;

int cnt;
bool isPrime[N];
int Prime[N];

void GetPrime(int n) {
    memset(isPrime, 1, sizeof(isPrime));
    isPrime[1] = 0;
    for (int i = 2; i <= n; ++i) {
        if (isPrime[i]) Prime[++cnt] = i;
        for (int j = 1; j <= cnt && i * Prime[j] <= n; ++j) {
            isPrime[i * Prime[j]] = 0;
            if (i % Prime[j] == 0) break;
        }
    }
}

int main() {
    int n, q;
    cin >> n >> q;
    GetPrime(n);
    while (q--) {
        int k;
        cin >> k;
        cout << Prime[k] << endl;
    }
    return 0;
}
```

28. 上述代码是在判断 k 是否是质数。[ ]
29. 如果输入的 $k(k \leq 10^5)$ 大于 n ，那么输出一定为 `0` 。[ ]
30. 数组 $Prime[i]$ 里的数一定是单调递增的。[ ]
31. 该程序的时间复杂度为[ ]。
    - A. $\mathcal{O}(n+q)$
    - B. $\mathcal{O}(n^2)$
    - C. $\mathcal{O}(nlogn)$
    - D. $\mathcal{O}(logn)$
32. 该程序中数组 $isPrime[i]$ 的作用是[ ]。
    - A. 判断i是否是质数

- B. 判断i是否是偶数
- C. 判断i是否是奇数
- D. 判断i是否是回文数

33. （4分）如果将程序中的 `if(i%Prime[j]==0) break;` 删去，则对程序造成的影响是[]。
    - A. 会改变程序的运行结果，但不改变程序的时间复杂度。
    - B. 会改变程序的运行结果，也会改变程序的时间复杂度。
    - C. 不会改变程序的运行结果，也不会改变程序的时间复杂度。
    - D. 不会改变程序的运行结果，但会改变程序的时间复杂度。

三、完善程序（单选题，每小题3分，共计30分）

阅读下面题目，完成第 $34 \sim 38$ 题。

题目描述

桌子上从左往右摆放着糖果，糖果是从左到右编号的，第 i 个糖果的权重是w_i，花椰妹和蒜头君吃糖果。
花椰妹可以从左边吃掉任意数量的糖果，她不能跳过糖果，她需要连续吃。
蒜头君可以从右边吃掉任意数量的糖果，他不能跳过糖果，他需要连续吃。
当然，如果花椰妹吃了某个糖果，那么蒜头君就不能吃了，反之亦然。
他们想要公平，他们的目标是吃同样重量的糖果，在满足这个条件下，请问他们总共最多能吃多少糖果？

输入说明

第一行一个正整数 T，表示有多少组测试数据。

对于每组测试数据，第一行一个数 n，表示桌子上有多少糖果。

对于每组测试数据，第二行 n 个数 $w_1, w_2, ..., w_n$，从左到右表示每个糖果的重量。

输出说明

对于每组测试数据，一个正整数，表示总共最多能吃多少糖果。

样例输入

```
4
3
10 20 10
6
2 1 4 2 4 1
5
1 2 4 8 16
9
7 3 20 5 15 1 11 8 10
```

样例输出

```
2
6
0
7
```

请补全下面的代码。

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

using namespace std;

const int N = 2e5 + 5;

int T, n;
int a[N], pre[N], suf[N];

int main() {
    cin >> T;
    while (T--) {
        cin >> n;
        for (int i = 1; i <= n; ++i)①;
        for (int i = 1; i <= n; ++i) pre[i] = pre[i - 1] + a[i];
        suf[n + 1] = 0;
        for (int i = n; i >= 1; --i)②;
        int ans = 0;
        for (int i = 1; i < n; ++i) {
            int l = i + 1, r = n, res = -1;
            while (l <= r) {
                int mid = ③;
                if (suf[mid] < pre[i]) r = mid - 1;
                else if (④) l = mid + 1;
                else {
                    res = mid;
                    break;
                }
            }
            if (res != -1) ans = max(ans, ⑤);
        }
        cout << ans << endl;
    }
    return 0;
}
```

34. ① 处应填
    - A. `cin>>a[i]`
    - B. `cout<<a[i]`
    - C. `cin>>a`
    - D. `cout<<a`
35. ② 处应填
    - A. `suf[i]=suf[i+1]+a[i]`
    - B. `suf[i]=suf[i-1]+a[i]`
    - C. `suf[i]=suf[i+1]-a[i]`
    - D. `suf[i]=suf[i-1]-a[i]`

36. ③ 处应填
    - A. `(l+r)>>2`
    - B. `(l+r)<<1`
    - C. `(l+r)>>1`
    - D. `(l+r)<<2`
37. ④ 处应填
    - A. `suf[mid]>=pre[i]`
    - B. `suf[mid]=pre[i]`
    - C. `suf[mid]<pre[i]`
    - D. `suf[mid]>pre[i]`
38. ⑤ 处应填
    - A. `i+n-res`
    - B. `i+n-res+1`
    - C. `i+res+1`
    - D. `i+res`

阅读下面题目，完成第 $39 \sim 43$ 题。

题目描述

定义 AND 为按位与操作，OR 表示按位或操作。
现在给定一个长度为 n 的数组 a 和一个非负整数 k，蒜头君最多可以执行 k 次以下类型的操作：选择一个 i，将 a_i 变为 $a_i \ OR \ 2^j$，其中 j 可以是 0 到 30 内的任意整数。
输出执行最多 k 次操作后，$a_1 \ AND \ a_2 \ AND ... AND \ a_n$ 的最大值。

输入说明

第一行一个正整数 T，表示有多少组测试数据。

对于每组测试数据，第一行两个数 n, k。

对于每组测试数据，第二行 n 个数，表示数组 a 的元素。

输出说明

对于每组测试数据，一个正整数，表示 $a_1 \ AND \ a_2 \ AND ... AND \ a_n$ 的最大值。

样例输入

```
4
3 2
2 1 1
7 0
4 6 6 28 6 6 12
1 30
0
4 4
3 1 3 1
```

样例输出

```
2
4
2147483646
1073741825
```

请补全下面的代码。

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

using namespace std;

#define ll long long

const int N = 2e5 + 5;
const int M = 35;

int T, n, k;
int a[N], num[M];

int main() {
    ①;
    while (T--) {
        ②;
        cin >> n >> k;
        for (int i = 1; i <= n; ++i) {
            cin >> a[i];
            for (int j = 0; j <= 30; ++j) {
                if (③) ++num[j];
            }
        }
        int ans = 0;
        for (int i = 30; i >= 0; --i) {
            ④;
            k -= n - num[i], ⑤;
        }
        cout << ans << endl;
    }
    return 0;
}
```

39. ① 处应填
    - A. `cin>>T`
    - B. `scanf("%d",T)`
    - C. `T=getchar()`
    - D. `gets(T)`
40. ② 处应填
    - A. `memset(num,0x7f,sizeof(num))`
    - B. `memset(num,1,sizeof(num))`
    - C. `memset(num,0,sizeof(num))`
    - D. `memset(num,0xcf,sizeof(num))`
41. ③ 处应填
    - A. `(a[i]>>j)|1`
    - B. `(a[i]>>j)&1`

- C. `(a[i]>>j)^1`
- D. `(a[i]>>j)<<1`

42. ④ 处应填
    - A. `if(k<=n-num[i]) continue`
    - B. `if(k>n-num[i]) continue`
    - C. `if(k<n-num[i]) continue`
    - D. `!if(k!=n-num[i]) continue`
43. ⑤ 处应填
    - A. `ans&=(1<<i)`
    - B. `ans|=(1<<i)`
    - C. `ans|=(1>>i)`
    - D. `ans&=(1>>i)`