# NY共克归艰（5）参考题解

## A-苹果派

略

```
#include <bits/stdc++.h>
using namespace std;
int main() {
  int a,b;
  scanf("%d %d", &a, &b);
  printf("%d", (a * 3 + b) / 2);
  return 0;
}
```

## B-花式复读

枚举轮数 $i$，如果第 $k$ 个被输出的数字在这一轮中，直接输出答案，否则进入下一轮。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100000 + 10;
int n, k, a[N];
int main() {
    scanf("%d%d", &n, &k);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    int now = 0;
    for (int i = 1; i <= n; i++) {
        if (k <= now + i && k >= now + 1) {
            return !printf("%d\n", a[k - now]);
        }
        now += i;
    }
}
```

## C-约数

- 若 $n$ 为 4 的倍数，令 $a = n/4, b = n/4, c = n/2$ 有解
- 若 $n$ 为 3 的倍数，令 $a = n/3, b = n/3, c = n/3$ 有解
- 否则一定无解。我们尝试反证：设存在解 $a = n/x, b = n/y, c = n/z$，那么 $\frac{1}{x} + \frac{1}{y} + \frac{1}{z} = 1$
  - 若 $min(x, y, z) \geq 5$ 那么 $\frac{1}{x} + \frac{1}{y} + \frac{1}{z} < 1$
  - 否则设 $x = 2$，那么 $\frac{1}{y} + \frac{1}{z} = \frac{1}{2}$，此时必有 $y, z \geq 5$，因此 $\frac{1}{y} + \frac{1}{z} < \frac{1}{2}$ 矛盾

```
#include <bits/stdc++.h>
using namespace std;
int n, t;
int main() {
    cin >> t;
    while (t--) {
        cin >> n;
        printf("%s\n", (n % 3 == 0 || n % 4 == 0) ? "Yes" : "No");
    }
}
```

## D-取球

一定存在一种最优决策是：先取球，再放球。取出的球一定是最左边的若干球和最右边的若干个球。枚举取出了哪些球，然后从小到大挑选出一些权值为负数的球扔掉。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 10002;
int n, K, v[N];
int main() {
    scanf("%d%d", &n, &K);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &v[i]);
    }
    int ans = 0;
    for (int i = 0; i <= n; i++) {
        for (int j = i + 1; j <= n + 1; j++) {
            int x = i + n - j + 1, score = 0;
            if (x > K)
                continue;
            vector<int> vec;
            for (int k = 1; k <= i; k++) vec.push_back(v[k]), score += v[k];
            for (int k = j; k <= n; k++) vec.push_back(v[k]), score += v[k];
            sort(vec.begin(), vec.end());
            for (int k = 0; k < min(K - x, (int)vec.size()); k++) {
                if (vec[k] < 0)
                    score -= vec[k];
            }
            ans = max(ans, score);
        }
    }
    cout << ans << endl;
}
```

## E-营养计划

简单的动态规划即可。

定义$dp[pos][x][y][z]$表示当前天数和三种能力值的方案数量。显然有

$dp[pos][x][y][z] = dp[pos+1][x+a[pos+1]][y+b[pos+1]][z+c[pos+1]] + dp[pos+1][x+d[pos+1]][y+e[pos+1]][z+f[pos+1]]$

答案为$dp[0][50][50][50]$

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

using namespace std;
typedef long long ll;
typedef unsigned long long ul;

const int mn = 100 + 1;
const int mod = 1e9 + 7;

int n;
int a[mn], b[mn], c[mn], d[mn], e[mn], f[mn];
int D[61][mn][mn][mn];

int Ans = 0;

void add(int &ans, int x) {
    ans += x;
    if (ans >= mod)
```

```
        ans -= mod;
}
int dp(int pos, int x, int y, int z) {
    if (x >= 100 || x <= 0 || y >= 100 || y <= 0 || z >= 100 || z <= 0)
        return 0;
    int &ans = D[pos][x][y][z];
    if (ans != -1)
        return ans;
    if (pos == n)
        return ans = 1;
    ans = 0;
    // do
    add(ans, dp(pos + 1, x + a[pos + 1], y + b[pos + 1], z + c[pos + 1]));
    // not do
    add(ans, dp(pos + 1, x + d[pos + 1], y + e[pos + 1], z + f[pos + 1]));
    return ans;
}
int main() {
    memset(D, -1, sizeof D);
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d%d%d%d%d%d", &a[i], &b[i], &c[i], &d[i], &e[i], &f[i]);
    }
    printf("%d\n", dp(0, 50, 50, 50));
    return 0;
}
```

## F-冒雨回家

动态规划枚举每种情况即可。

定义$dp[pos][time]$表示还剩的距离和花费的时间。在合法转移的前提下，有：

$dp[pos][time] = \min\{dp[pos-1][time+1] + a[time]/3, dp[pos-2][time+1] + a[time]/2, dp[pos-3][time+1] + a[time], dp[pos][time+$

答案即为$dp[n][1]$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int mn = 1e4 + 5;
int n, m;
int a[mn];
int d[1005][mn];
int dp(int pos, int time) {
    if (pos <= 0) return 0;
    int &ans = d[pos][time];
    if (ans != -1) return ans;
    ans = 1e9;
    if (time > m) return ans;
    // to 1 meter
    ans = min(ans, dp(pos - 1, time + 1) + a[time] / 3);
    // to 2 meter
    ans = min(ans, dp(pos - 2, time + 1) + a[time] / 2);
    // to 3 meter
    ans = min(ans, dp(pos - 3, time + 1) + a[time]);
    // wait
    ans = min(ans, dp(pos, time + 1));
    //printf("%d %d %d\n", pos, time, ans);
    return ans;
}
int main() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) cin >> a[i];
    memset(d, -1, sizeof d);
```

```
    cout << dp(n, 1) << "\n";
    return 0;
}
```