A. Чук и Гек

Нужно посчитать количество лишних конфет для Чука и Гека. Если разрешено провозить C килограмм, то это значит, что разрешено провозить C * 1000 грамм. Поделим это число на количество конфет с округлением вниз и мы получим количество конфет, которые можно провезти. Ответ не может быть отрицательным – это нужно учесть.

Решение на языке Python 3

```
n,  m = map(int,  input().split())
a,  b,  c = map(int,  input().split())
print( max(0,  n - c * 1000 // a), max(0,  m - c * 1000 // b)  )
```

Решение на языке GNU C++11

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    int a, b, c;
    cin >> a >> b >> c;
    cout << max(0, n - c * 1000 / a) << " " << max(0, m - c * 1000 / b);
}
```

B. Дед Мазай

В задаче требуется организовать ввод данных с использованием цикла while до тех пор, пока на вход не будет подано число 0. Сумма переданных на вход чисел – это ответ.

Решение на языке Python 3

```
s = 0
while True:
    x = int(input())
    if x == 0:
        break
    s += x
print(s)
```

Решение на языке GNU C++11

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ios::sync_with_stdio(0);
    int s = 0, x;
    do {
        cin >> x;
        s += x;
    } while (x != 0);
    cout << s;
}
```

C. Список дел

Задача решается жадным алгоритмом – все дела сортируются по возрастанию времени, которое необходимо на них потратить. Затем выбирается наибольшее количество дел из начала этого списка так, чтобы сумма их времени не превосходила заданной продолжительности рабочего дня.

Решение на языке Python 3

```
n,  m = map(int,  input().split())
a = [int(input()) for i in range(m)]
n *= 60
a.sort()
for i in range(m):
    if a[i] > n:
        break
    n -= a[i]
print(i,  n)
```

Решение на языке GNU C++11

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ios::sync_with_stdio(0);
    int n, m;
    cin >> n >> m;
    vector<int> a;
    for(int i=0; i<m; ++i)
    {
```

```
        int x;
        cin >> x;
        a.push_back(x);
    }
    sort(a.begin(), a.end());
    n *= 60;
    int i = 0;
    while(i < m && a[i] <= n)
    {
        n -= a[i];
        ++i;
    }
    cout << i << endl << n << endl;
}
```

D. Сыродел Василий

Нужно сложить все дроби. На языке Python это можно сделать, например, с использованием класса Fraction. Универсальное решение – это посчитать сумму числителей, а также посчитать знаменатель, как произведение всех знаменателей без общих делителей (делим на НОД). Просто произведение всех знаменателей может привести к переполнению.

Решение на языке Python 3

```
from math import gcd
n = int(input())
frac = []
p = 1
for i in range(n):
    a,  b = map(int,  input().split("/"))
    frac.append( (a, b) )
    if p % b != 0:
        p *= b // gcd(p,  b)
frac_top = 0
for f in frac:
    frac_top += f[0] * (p // f[1])
print(frac_top // p,  "{:d}/{:d}".format( (frac_top % p) // gcd(frac_top,  p),  p // gcd(frac_top,  p) ))
```

Решение на языке GNU C++11

```
#include<bits/stdc++.h>
using namespace std;
long long gcd(long long a, long long b)
```

```
{
    if (b == 0) return a;
    return gcd(b, a % b);
}
int main()
{
    int n;
    long long frac_bot = 1LL;
    vector< pair<int, int> > fractions;
    scanf("%d", &n);
    for(int i=0; i<n; ++i)
    {
        int x, y;
        scanf("%d/%d", &x, &y);
        fractions.push_back( {x, y} );
        if (frac_bot % y != 0) frac_bot *= 1LL * y / gcd(frac_bot, y);
    }
    long long frac_top = 0;
    for(auto frac: fractions)
    {
        long long d = frac_bot / frac.second;
        frac_top += 1LL * frac.first * d;
    }
    cout << frac_top / frac_bot << " ";
    frac_top = frac_top % frac_bot;
    long long d = gcd(frac_top, frac_bot);
    cout << frac_top / d << "/" << frac_bot / d << endl;
}
```

E. В траве сидел кузнечик

Задача решается с использованием метода динамического программирования. Для этого можно использовать массив dp, в котором хранить все прошлые значения, но достаточно трех переменных. Рекуррентная формула для ДП следующая:

$$dp[i] = \begin{cases} a[0], & i = 0 \\ max\begin{pmatrix} dp[i-3] - dist[i-3], \\ dp[i-2] - dist[i-2], \\ dp[i-1] - dist[i-1] \end{pmatrix} + a[i], & i > 0 \end{cases}$$

a[i] – количество травы на кочке, dist[i] – расстояние между кочками i и i+1. Важно отдельно проверить, что кочка достижима, т.е. что максимальная энергия при достижении кочки i из рассматриваемых вариантов не является отрицательной.

Решение на языке Python 3

```
n = int(input())
a = [int(x) for x in input().split()]
x = list(map(int,  input().split()))
dp = [-1] * 3
dp[0] = a[0]
for i in range(1,  n):
    e = -1
    for j in range(1,  min(3, i) + 1):
        e = max(e,  dp[(i-j) % 3] - x[j-1])
    dp[i % 3] = e
    if dp[i % 3] > -1:
        dp[i % 3] += a[i]
print(dp[(n - 1) % 3])
```

Решение на языке GNU C++11

```
#include<bits/stdc++.h>
using namespace std;

int n;
vector<int> grass;
vector<int> dist;
vector<int> dp;

void solve()
{
  dp.resize(n, -1);
  dp[0] = grass[0];
  for(int i=1; i<n; ++i)
  {
    dp[i] = -1;
    for(int j=1; j <= min(3, i); ++j)
    {
      dp[i] = max(dp[i], dp[i-j] - dist[j-1]);
    }
    if (dp[i] > -1) dp[i] += grass[i];
  }

  cout << dp[n-1];
}

int main()
{
    cin >> n;
    for(int i=0; i<n; ++i)
    {
      int x;
      cin >> x;
      grass.push_back(x);
    }
```

```
    for(int i=0; i<3; ++i)
    {
      int x;
      cin >> x;
      dist.push_back(x);
    }
    solve();
    cout << endl;
}
```