#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\*\* split\_string(char\*);

// Complete the chocolateFeast function below.

int chocolateFeast(int n, int c, int m) {

int num\_wrappers,ans;

num\_wrappers = n/c;

ans = n/c;

int a,b;

while(num\_wrappers >= m)

{

a=num\_wrappers / m;

b = num\_wrappers % m;

ans = ans + a;

num\_wrappers = b + a;

}

return ans;

}

int main()

{

FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

char\* t\_endptr;

char\* t\_str = readline();

int t = strtol(t\_str, &t\_endptr, 10);

if (t\_endptr == t\_str || \*t\_endptr != '\0') { exit(EXIT\_FAILURE); }

for (int t\_itr = 0; t\_itr < t; t\_itr++) {

char\*\* ncm = split\_string(readline());

char\* n\_endptr;

char\* n\_str = ncm[0];

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

char\* c\_endptr;

char\* c\_str = ncm[1];

int c = strtol(c\_str, &c\_endptr, 10);

if (c\_endptr == c\_str || \*c\_endptr != '\0') { exit(EXIT\_FAILURE); }

char\* m\_endptr;

char\* m\_str = ncm[2];

int m = strtol(m\_str, &m\_endptr, 10);

if (m\_endptr == m\_str || \*m\_endptr != '\0') { exit(EXIT\_FAILURE); }

int result = chocolateFeast(n, c, m);

fprintf(fptr, "%d\n", result);

}

fclose(fptr);

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) { break; }

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

size\_t new\_length = alloc\_length << 1;

data = realloc(data, new\_length);

if (!data) { break; }

alloc\_length = new\_length;

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

}

data = realloc(data, data\_length);

return data;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces - 1] = token;

token = strtok(NULL, " ");

}

return splits;

}