#include <assert.h>

#include <ctype.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\* ltrim(char\*);

char\* rtrim(char\*);

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

int parse\_int(char\*);

/\*

 \* Complete the 'arrayManipulation' function below.

 \*

 \* The function is expected to return a LONG\_INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER n

 \*  2. 2D\_INTEGER\_ARRAY queries

 \*/

long arrayManipulation(int n, int queries\_rows, int queries\_columns, int\*\* queries) {

long \*arr=calloc(n+2,sizeof(long));

for(int i=0;i<queries\_rows;i++){

    int a=queries[i][0];

    int b=queries[i][1];

    int k=queries[i][2];

    arr[a]+=k;

    arr[b+1]-=k;

}

long max=LONG\_MIN;

long current=0;

for(int i=1;i<=n;i++){

    current+=arr[i];

    if(current>max){

        max=current;

    }

}

free(arr);

return max;

}

int main()

{

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

    char\*\* first\_multiple\_input = split\_string(rtrim(readline()));

    int n = parse\_int(\*(first\_multiple\_input + 0));

    int m = parse\_int(\*(first\_multiple\_input + 1));

    int\*\* queries = malloc(m \* sizeof(int\*));

    for (int i = 0; i < m; i++) {

        \*(queries + i) = malloc(3 \* (sizeof(int)));

        char\*\* queries\_item\_temp = split\_string(rtrim(readline()));

        for (int j = 0; j < 3; j++) {

            int queries\_item = parse\_int(\*(queries\_item\_temp + j));

            \*(\*(queries + i) + j) = queries\_item;

        }

    }

    long result = arrayManipulation(n, m, 3, queries);

    fprintf(fptr, "%ld\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;

        }

        alloc\_length <<= 1;

        data = realloc(data, alloc\_length);

        if (!data) {

            data = '\0';

            break;

        }

    }

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

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

        data = realloc(data, data\_length);

        if (!data) {

            data = '\0';

        }

    } else {

        data = realloc(data, data\_length + 1);

        if (!data) {

            data = '\0';

        } else {

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

        }

    }

    return data;

}

char\* ltrim(char\* str) {

    if (!str) {

        return '\0';

    }

    if (!\*str) {

        return str;

    }

    while (\*str != '\0' && isspace(\*str)) {

        str++;

    }

    return str;

}

char\* rtrim(char\* str) {

    if (!str) {

        return '\0';

    }

    if (!\*str) {

        return str;

    }

    char\* end = str + strlen(str) - 1;

    while (end >= str && isspace(\*end)) {

        end--;

    }

    \*(end + 1) = '\0';

    return str;

}

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;

}

int parse\_int(char\* str) {

    char\* endptr;

    int value = strtol(str, &endptr, 10);

    if (endptr == str || \*endptr != '\0') {

        exit(EXIT\_FAILURE);

    }

    return value;

}