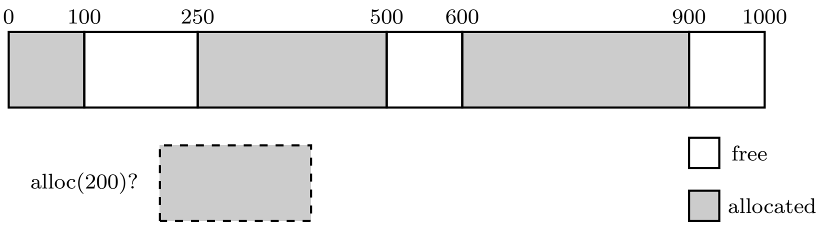
Assignment #3 Linked List
Problem 2. Memory Management

Score: 60 points

Due: Oct 28, 2020

Have you ever wondered how `malloc()` works internally? Memory management is one of the famous problems in computer science. Memory can be thought of as a sequence of bytes, and memory allocation is to find a contiguous memory area of the requested size from the unused space. After a lot of memory allocations and deallocations are repeated, the memory area will be divided into little pieces of different sizes, which is called *fragmentation.*



The above figure shows how memory space can be wasted by fragmentation. A 1,000 bytes memory space consists of 3 allocated memory areas and 3 free memory areas. When a user requests 200 bytes of memory, the memory cannot be allocated even though the total amount of free space(350 bytes) is larger than this. Therefore, the goal of memory management is to mitigate fragmentation so that the memory space can be used efficiently.

While real world memory allocator is implemented with more complex algorithms, in this assignment, you will implement the simple memory allocator with following strategy to handle memory allocation requests. You need to implement `alloc` and `free` operations.

- alloc(n): Allocate n bytes of memory and return the starting address of the allocated area. The following is the detailed algorithm.
  1. Find all available free contiguous memory areas with a size of n bytes or more.
  2. Pick the smallest memory area among these.
  3. If there are several, choose the one with the smallest memory address,
  4. Then, allocates n bytes from the beginning of the memory area and make the rest as a smaller free memory area.
  5. If allocation is not possible during the above process, return -1.
- free(m, n): Deallocate the n bytes of allocated memory with starting address of m. The following is the detailed algorithm.
  1. Returned memory area becomes free memory space.
  2. If there are adjacent free memory areas in previous or after, they are merged into one larger free area.

Input

The first line contains two integers L ($1 \le L \le 10^6$) and N ($1 \le N \le 10^4$) — the total length of the given memory area and the number of user requests.

Each of the next N lines describes one of the following operations.

- `alloc n`: allocate n bytes memory area.
- `free m n`: deallocate the memory area of size n bytes and starting address of m.

It is guaranteed that the memory size and address in given inputs are fit in the memory space. The memory area of `free` request is guaranteed to be allocated previously.

Output

For each `alloc` request, print the single line with the starting address of alloacted memory. If allocation is not possible, print `-1`.

Sample Inputs

Sample Input 1	Sample Output 1
200 4 alloc 100 alloc 30 alloc 70 alloc 50	0 100 130 -1

Sample Input 2	Sample Output 2
1000 10 alloc 50 alloc 100 free 0 50 alloc 80 alloc 30 free 50 100 alloc 200 free 150 80 alloc 200 alloc 600	0 50 150 0 230 30 -1

Grading Policy

Submissions will be graded based on the number of test cases passed. There are execution time and memory limitations, so be sure to write efficient code. Submit your code on goormEDU (`https://skku.goorm.io`).

For late submission, grades will be deducted by 25% per late day. That is, after 4 days, the grade will be zero.

The assignment should be done by yourself. We use plagiarism detection tool on all submissions.

If you have any questions, leave them in the spreadsheet or send an email to the TA. Please do not use i-Campus message. (TA Youngjae Lee: `yjlee4154@gmail.com`)

QA Spreadsheet Link
`https://docs.google.com/spreadsheets/d/1RDbEqFMS1FUO-KHcqX3BvhMHNJI0Y-wpOPXAmE-zdQU/edit?usp=sharing`