CS3300: Compiler Design **Odd Sem**

Lab #14

Memory Manager Deadline: 11/11/2018, 11:55PM

Task

Implement a simple memory manager in C/C++. The memory manager should:

- Process memory allocation (`malloc`) and deallocation (`free`) requests
- Keep track of leaked memory
- Support various memory allocation strategies.

`malloc` and `free` requests to the memory manager will be given through `stdin`.

To perform allocations quickly, you have to maintain a list of free chunks internally and then, upon a memory allocation request, choose one free chunk from the list. The memory allocations should always be aligned with the left boundary of a free chunk.

The memory manager should not coalesce free chunks, that is, when an allocated memory is freed, it becomes a new separate free chunk and does not coalesce with the adjacent free chunks, if there are any. Any subsequent allocation requests that cannot fit inside any one free chunk should be rejected.

Memory Allocation Strategies

The memory manager should support the following allocation strategies:

- First Fit
- Best Fit
- Worst Fit

During an allocation, if more than one free chunk best-fits/worst-fits the requested memory, you have to choose the chunk that comes first, i.e., the one which has the least base address.

Memory Leakage

A memory chunk is leaked if it is allocated but not freed by the end of the program. You have to keep track of all such chunks and report it in the output.

Input

The input is provided in `stdin`. The first line contains two parameters - the total memory available N, and the allocation strategy to use. The allocation strategy is one of `first-fit`, `best-fit` and `worst-fit`. Note that the bytes in the available memory are addressed from 0 to $N-1$.

Each of the subsequent R lines contain either a `malloc` or a `free` request. Each malloc request has the format `malloc <pointer-variable-name> <object size>` and each free request has the format `free <pointer-variable-name>`

Output

The output should be written to `stdout`. The enitre output consists of two blocks separated by an empty line. The first block contains R lines, where the ith line contains the status of the ith `malloc`/`free` request. The second block starts with a line containing the string `leaked-memory`, and then followed by a list of all leaked chunks, one per line, in ascending order of their start addresses. For every leaked chunk, you have to print the chunk's start address and end address.

For every `malloc` request, print one of the following status:

1. `success`, if the request succeeded
2. `out-of-memory error`, if there is no big enough free chunk that can be used for allocation

For every `free` request, print one of the following status:

1. `success`, if the request succeeded
2. `invalid-pointer error`, if the pointer in the request has never been allocated memory through a successful `malloc` request.
3. `double-free error`, if the pointer in the request has already been freed. In this case, don't deallocate any block.

Constraints

- Total memory available $N : 0 \leq N \leq 10^5$
- Number of requests $R : 0 \leq R \leq 10^3$
- Object size in any malloc request > 0

Examples

Example 1 - First Fit

Input

```
100 first-fit
malloc p1 23
malloc p2 34
malloc p3 80
free p2
```

Output

```
success
success
out-of-memory error
success

leaked-memory
0 22
```

Example 2 - Best Fit

Input

```
10 best-fit
malloc p1 2
malloc p2 3
free p1
malloc p3 2
free p1
free p5
malloc p3 4
malloc p4 1
free p3
free p2
```

Output

```
success
success
success
success
double-free error
invalid-pointer error
success
success
success
success

leaked-memory
0 1
9 9
```

Submission

Submit a tar.gz file with filename as <ROLLNO >.tar.gz (eg. CS12B043.tar.gz) containing the following structure:

- CS12B043 <directory>
  - *.c/*.cpp
  - Makefile

The Makefile should generate an executable **a.out**.