Goal
Study the influence of different heap management options.

Problem Statement
Sponge Bob, Patrick, and Squidward each have a favorite technique for managing the heap (they all agree that Mr. Crabs stacks are to die for). Sponge Bob asserts that all memory blocks should be the same size, so he wrote his own constant-size allocator (well with your help that is), Patrick likes the one that is built into Sea, and Squidward prefers for someone else to take out the garbage (we will let Java to this).

Analysis (What is the client’s problem)
[Software Engineers first construct an analysis (what is it) and then a design (how will it be done).]
[Here is an example question that an engineer might ask. Can you think of others?]
Q: What measure should be used to compare the three heap management techniques?
A: At least CPU time (check out /usr/bin/time).
[Commit your analysis (as part of README.md) before moving on to the design.]

Design (How will this problem be solved)
In this case the big how question is “how will you and Sponge Bob maintain your custom heap?” Possible answer to this question involve lists, stacks, or perhaps trees.
[Commit your design (as part of README.md) before moving on to the plateau schedule.]

Plateau Schedule
Before you start coding take the time to write out a build plan.
[Commit your plateau schedule (as part of README.md) before moving on to the code.]

What to hand in (deadline [points]) (Please no .docx files.)
(1) 6am 3/15 [0]. I will pull your analysis and then push comments.
(2) 6am 3/17 [20]. I will pull your design and then push comments.
(3) class time 3/24 [5]. A well-formatted 2-up printout of your source code. You must use a2ps after removing all the tabs from your code. Indent code 2 or 4 spaces at most.
(4) class time 3/24 [75]. A GitHub repo that includes (you must use these names as the grading script will assume their use!)
   • README.md with the sections Analysis, Design, Plateau Schedule, and My Analysis (i.e., your take on the generated data).
   • main.c and JavaWay.java, and
   • Makefile (where make all and make test will build and test your code).
Assignment Notes

• You must use `#define` macros for the two heap functions `get_mem()` and `free_mem()`.

• You must use `#ifdef` to enable the compile-time selection between Sponge Bob’s way and Patrick’s way as required by the Makefile. The `#ifdef` should guard definitions of `get_mem()` and `free_mem()` using one of the two approaches.

• By special permission during the season of Lent, your code can include static (file scope) variables.

• Here is the test algorithm to use
  
  create an array of BLOCK_COUNT entries  
  for ITERATIONS iterations  
  pick a random block  
  if that block is allocated, free it  
  otherwise allocate BLOCK_SIZE bytes and assign it to the block

• Target a run time for Patrick’s version of about 5 to 10 seconds (by setting ITERATIONS).

• Better answer will experiment with the impact of BLOCK_COUNT and BLOCK_SIZE as well as any initialization of Sponge Bob’s data structures (consider using command-line arguments).

• I expect to pull your code, run make, and then run my test script.

• You must use memwatch with all C and Sea code. Better answers will report unfreed bytes 0.

• The output of `git log` will again factor into your grade.

• Include under the heading My Analysis the maximum memory usage for each of the three using a common configuration. Repeat this comparison for a few other configurations. I used  

  `top -d 1 | grep "java\|_way"`

running in a separate window. Format this information nicely!!

• Run the final time trials without memwatch! (e.g., edit the Makefile; make clean; make all; make test.)

• Here is the `git` classroom invitation  

  https://classroom.github.com/assignment-invitations/999aa673b73a752a1ea6d68f531e1cbf

• Want to have more fun? Plankton’s allocator includes “`#define free_mem(x)`” because freeing memory is a waste of time. Is he right?