

CS 2213, Spring 2009
Assignment 3: Towers of Hanoi
Due: 5pm, Friday, February 13, 2009

Goal: The purpose of this assignment is to learn to write relatively multi-file multi-function C programs with recursion.

Objectives: be able to write multi-file, multi-function C programs; be able to understand and utilize recursive functions; and be able to use external variables in a C program.

Permissible Collaboration and Resources. This assignment is to be completed *individually*. You are allowed to discuss general strategies for solving assignments with fellow students or other individuals, but it is no longer a 'general strategy' if the discussion gets to the level of detail of what would be done in actual lines of code found in these programs and discussions of 'general strategy' should not be taking place while either party is editing their source code. (It is, however, perfectly acceptable to discuss C language constructs and concrete examples of them which are not taken directly from anyone's solution to an assignment.) In addition, you may seek more specific help from mentors and lab tutors in trying to understand why their code doesn't work. Furthermore, you may also seek help and guidance of any kind from the TA and instructor. *You may not look at any code for solving the Tower of Hanoi* other than what is contained in this document or included in other material/links provided by the instructor or TA. In particular, although you may look at the wikipedia article for Tower of Hanoi, you are **not** allowed to consult the web sites linked to from Wikipedia or found through google, especially pages that contain code for solving the Tower of Hanoi.

Towers of Hanoi Problem

The towers of Hanoi is puzzle consisting of some number n (often 8) disks each of different sizes that can be moved among three different (possibly empty) stacks (let's call them A, B, and C). Initially, all of the disks are stacked on stack A in order from the largest at the bottom to the smallest at the top. The goal is to move all of the disks to stack C subject to the following rules:

1. only one disk can be moved at a time,
2. that disk can only be moved from the top of one stack to the top of another stack, and
3. the disk be smaller than any disks that it is placed on top of.

For $n = 1$, the solution is trivially, and consists of just moving the disk directly from the source A to the destination C .

For $n = 2$, the solution requires using the third stack B as temporary storage to get the top (small) disk out of the way of the bottom (large) disk. First move the

top (small) disk from A to B . Then move the large disk from A to C . The finally move the small disk from B to C .

In general, this puzzle can be solved inductively to obtain a recursive procedure for moving n disks from a *source* stack to a *destination* stack using a third *intermediate* stack as intermediate storage. The base case is $n = 1$ as described above. For the inductive case, start by assuming that you have a procedure that can be used to solve the problem of moving the top $n - 1$ from *source* to *destination* via *intermediate*. Then you can move n (where $n > 1$) disks from X to Z via Y by:

1. move the $n - 1$ smallest disks from X to Y
2. move 1 disk (the largest) from X to Z
3. move the $n - 1$ smallest disks from Y to Z

Note that this procedure works, even if there are additional disks already on X , Y , or Z as long as they are all bigger than the n disks being moved around.

For further information, including an animation showing how the puzzle is solved, you may read Wikipedia's article on the Tower of Hanoi: http://en.wikipedia.org/wiki/Tower_of_Hanoi. **But you are not allowed to follow the links from the wikipedia article to additional code besides what is directly included as part of the article.**

Implementing a Solution to Towers of Hanoi

Implement a solution to the Towers of Hanoi problem as a C program using a recursive function.

Your C program should be split into four files. The first, `main.c`, should consist of a `main()` function. The second, `hanoi.c` should contain a function `hanoi()`. The third `move.c` should contain a `make_move()` function. Put prototypes for each of the functions, declarations for shared external variables, along with an `enum stack` for indicating the three stacks { A , B , C } symbolically in `hanoi.h`.

The `main()` function should use `getchar()` to input a number n , call the function `hanoi()` function to move n disks from stack A to stack C , and then print out how many moves were needed to solve the n -disk hanoi problem. Reading n can be accomplished as follows: initialize n to 0. In a loop, use `getchar()` to read in each character. If the character is between '0' and '9' subtract '0' to get an integer d representing the value of that character, and update n by multiplying n by 10 and adding d to that. When you reach a character (including EOF) that is not a digit, you are done; n will contain the number represented by that sequence of digits.

`void hanoi(int n, enum stack src, enum stack dst)` should be a recursive function that solves the problem of moving n discs from stack `src` to stack `dst` as outlined above. If you have trouble figuring out how to do this, you may wish to consult the Pascal function found in the Wikipedia article, but make sure you understand how it works! (Additional hint: Since `enum` symbols A , B , C represent unique integers, to figure out the third stack via from `src`, `dst`, you can rely on the algebraic equality: $src + dst + via = A + B + C$.) For each move that `hanoi()` needs to make, it should call the function `void make_move(enum stack source, enum stack dest)` in `move.c`.

Each time `void make_move(enum stack src, enum stack dst)` is called it should use `printf` to output a line of text of the form:

```
Move #25: B -> A
```

Use/update an external variable `number_of_moves` to track the number of moves made so far. Create the correct declaration for it in `hanoi.h` so that it can also be accessed by the `main()` function to report how many moves were needed. You may also wish to define a helper function for `make_move()` to convert from the enum stack to the characters 'A', 'B', or 'C' using a switch statement.

Submission. Please create a README file divided into three sections: ANSWERS TO QUESTIONS, NOTES TO GRADER, and CERTIFICATION.

1. Under the heading ANSWERS TO QUESTIONS. Please answer the following questions:
 - Please explain why your `hanoi()` function is guaranteed to eventually terminate for any positive value of n .
 - If it takes k moves to solve `hanoi($n - 1$)`, how many moves will it take to solve `hanoi(n)`?
 - What single gcc command would you use to compile (and link) your source files into a program named `hanoi`?
 - What sequence of gcc commands would you use to compile each of the individual compilation units (.c files) into object (.o) files, and then links those object files into a program named `hanoi`?
2. Under NOTES TO GRADER, add any notes, you wish the TA/instructor to consider while grading.

In particular, please document any known problems with your program. If gcc produces any warning or error messages (with the switches: `-std=c99 -pedantic -Wall -Wextra -Wstrict-prototypes -Wno-unused-parameter`), please list them, describe what you think they mean, and say whether you think it indicates a problem in your code or if gcc is just being 'too picky.'
3. Under CERTIFICATION, please answer the following two questions in your README file:
 - (a) Did you use any code/materials other than those provided by or referenced by the instructor or TA in completing this assignment? If so, please cite the source (including its URL if it is online) and the nature and extent of your use of that material.

If such material is prohibited by the directions for this assignment, points may be deducted, but as long as such use is completely disclosed in the README file (including source, nature, and extent), it will not be considered to constitute scholastic dishonesty.
 - (b) Did you work with anyone or get help in completing this assignment from anybody other than the instructor or TA? (This includes mentors and tutors.) If so, please describe the nature and extent of the help received.

If such help or collaboration exceeds what is allowed for this assignment according to the directions above, points may be deducted, but

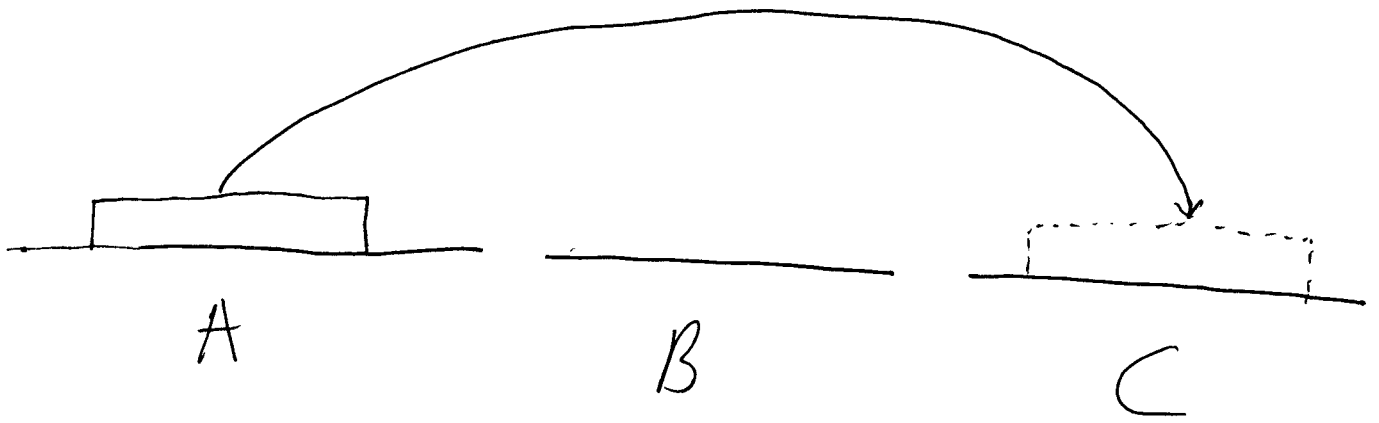
as long as the help or collaboration is completely disclosed in the README file, it will not be considered to constitute scholastic dishonesty.

Below your answer to those two questions, include the following statement:
I certify that except as noted above, the work I submit for this assignment has been completed solely by me without any outside help and without looking at any code for solving the Towers of Hanoi problem other than what is contained in this document or included in other material/links provided by the instructor or TA.

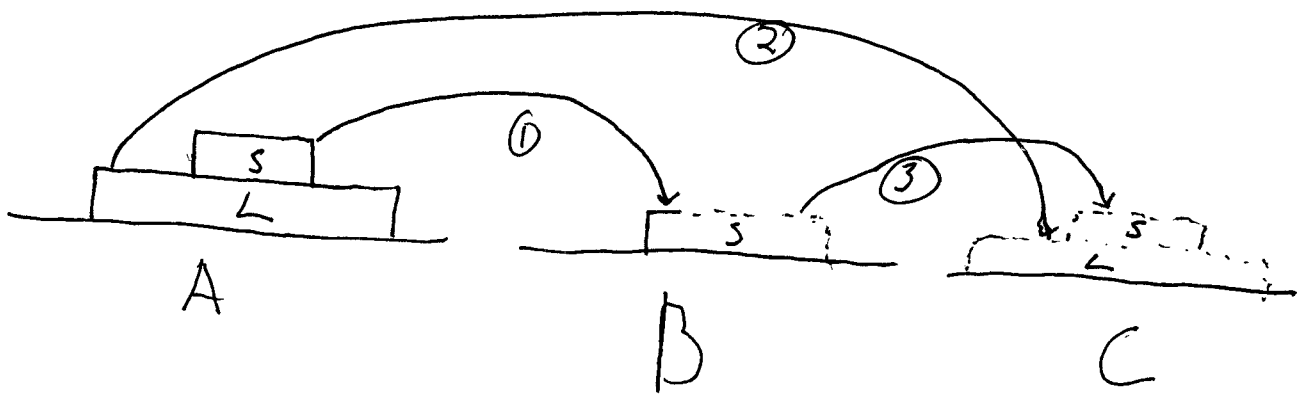
Below this statement, please write your name and date.

Use svn to commit your final submission by 5pm on the due date.

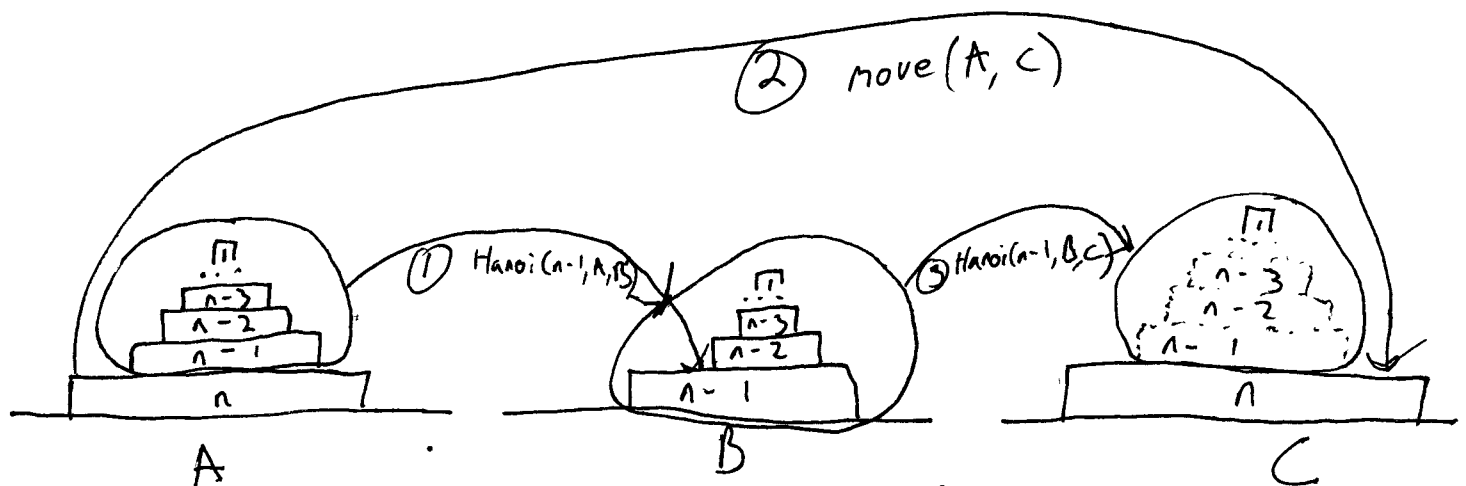
1. Make sure that you have all of the required files and they've all been added to svn. The expected contents of `assgn3` directory in the repository includes: `assignment3.pdf`, `README`, `hanoi.h`, `main.c`, `hanoi.c`, `move.c`. It is a also good idea to view your repository with your web browser to make sure all of the files are there and contain the most up-to-date contents.
2. You can continue to make changes, and re-commit as often as you want. The last version committed before 5pm on the due date will be graded.



Hanoi (1, A, C)



Hanoi (2, A, C)



Hanoi (n, A, C)