

CS 2213, Spring 2009  
Assignment 4: Towers of Hanoi Redux  
Due: 5pm, Friday, February 27, 2009

*Goal:* The purpose of this assignment is to learn to use and manipulate pointer, two-dimension arrays, and strings.

*Objectives:* be able to write code that navigates and extracts the string for the first command-line argument stored in `argv`; be able to write code that uses `sscanf()` to parse an integer found in a command-line argument; be able to write code to declare and use a two-dimensional array; be able to write code that accesses the row-element of a two dimensional array as a 1-dimensional array; be able to write code that uses pointer arithmetic to maintain a stack discipline.

*Relation to Assignment 3.* This assignment will build on your solution to Assignment 3. If you did poorly on Assignment 3, you have two options:

1. Fix the problems you had with Assignment 3. When you submit Assignment 4, note in your README the corrections you've made. You will be additional bonus points for Assignment 4 of up to half the points you lost for those issues on Assignment 3.
2. Alternatively, you may request a solution to Assignment 3 by sending an email to the TA and instructor. A solution to Assignment 3 will be committed to the `assgn4` directory of your subversion repository. If you have requested a deadline extension for Assignment 3, you will not be able to request a working solution until after the extended deadline has passed.

### **Towers of Hanoi with Explicit Representation of Disks**

In this assignment you will extend your program from Assignment 3 so that:

1. it gets the number of disks from the command-line instead of the standard input,
2. it prints a message out each time `hanoi()` gets called showing `hanoi()`'s parameters, and
3. when the number of disks is less than 10, it will keep track of the location of the different sized disks and print out a textual representation of the towers after each move.

Running the modified program with a command-line argument of 3 (`./hanoi 3`) should result in output that looks like:

Initial Sate:

A: 321

B:

C:

[Solving Hanoi(3, A, C)]

```

[Solving Hanoi(2, A, B)]
[Solving Hanoi(1, A, C)]
Move #1: A -> C
  A: 32
  B:
  C: 1
Move #2: A -> B
  A: 3
  B: 2
  C: 1
[Solving Hanoi(1, C, A)]
Move #3: C -> A
  A: 3
  B: 21
  C:
Move #4: A -> C
  A:
  B: 21
  C: 3
[Solving Hanoi(2, B, C)]
[Solving Hanoi(1, B, A)]
Move #5: B -> A
  A: 1
  B: 2
  C: 3
Move #6: B -> C
  A: 1
  B:
  C: 32
[Solving Hanoi(1, A, C)]
  A:
  B:
  C: 321

```

## Implementation

1. After updating your subversion repository to get the assign4 directory, copy the following files from assign3 into your assign4 directory: `hanoi.h`, `main.c`, `hanoi.c`, and `move.c`.
2. Create a makefile to automate compiling your program using explicit rules. The Makefile should have a rule for compiling each of the source files into an object file and a rule for linking the object files together to produce an executable called `hanoi`.
3. Modify the `main()` function to read the number of disks from its command-line arguments (i.e., `argc`, and `argv` instead of the standard input. Specifically, you should remove the code for looping over the result of `getchar()` and converting that to a number. Instead, you should follow the

argv pointer to a string representing the first command-line argument and then parse that string using the function `sscanf()` to obtain an integer representing the number of disks.

Note: Your program should be robust against invalid input. If there is an incorrect number of command-line arguments (0 or more than 1), your program should not crash, but rather give an error message and exit gracefully if it is unable to continue. Likewise, if the command-line argument is not a number.

4. Modify your `hanoi.h/move.c` to create an external char array `tower_char[]` initialized so that `tower_char[A]` evaluates to 'A', `tower_char[B]` evaluates to 'B', and `tower_char[C]` evaluates to 'C'.

5. Modify your `hanoi()` function so that each time it is called, it will output text of the form:

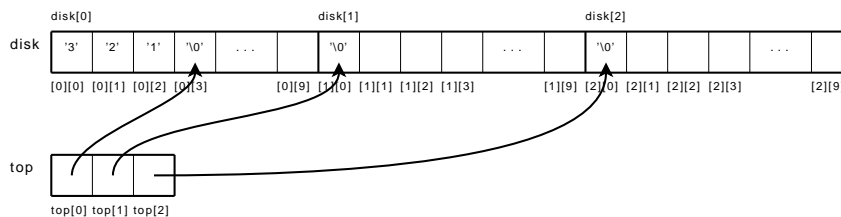
[Solving Hanoi(3, A, C)]

You can use `tower_char` to convert the second and third parameters of `hanoi()` into characters that can be printed with `printf()`.

6. Modify `main.c`, `move.c`, and `hanoi.h` to maintain and update a data structure storing what disks are currently found in the each of the towers A, B, and C. This only needs to be done if there are less than 10 disks. (Use a `#define` to create a symbolic constant `MAX_DISKS` to represent the limit of 10 disks.) If there are more than 10 disks the program should revert to just listing the moves as in assignment 3.

- (a) The disk data structure is probably best declared as a 2-dimensional array of characters where the outer dimension (row) index selects among the three towers and the inner dimension (column) index selects among disks in order from the base to the top of that tower. The contents of each character element (at positions less than the height of the tower) should represent the size of that disk. You may also want to use the characters '1', '2', ... '9' to represent the different sizes of disks in the array and to leave an extra element for a null-character so that you can use the inner array directly as a string (i.e., make `disk` `MAX_DISKS+1` characters in each row). This `disk` data structure can be declared as `static` external variable within the `move.c`.

You should also declare a `static` external array, `top`, that contains pointers that allows one to access/modify the top element. More precisely, you'll want to point at the null-character one element after the character for the top disk so you have something to point at when the tower is empty. You will use pointer arithmetic as you push and pop disks in order to maintain these pointers so that `top[t]` always points to `disk[t][n]` where `n` is the current height of tower `t`.



You may also need to add an additional external variable holding the number of disks so that the new code can be suppressed when this value is less than MAX\_DISKS

- (b) Create the following helper functions in `move.c` to initialize the data structure, pop one disk off of the top of a particular tower, push one disk on to the top of a particular tower, pop one disk off of a particular tower, and print out a textual representation of the current states of all three towers, respectively:

```
void init_towers(int number_disks);
void push(enum stack tower, char disk_size);
char pop(enum stack tower);
void print_towers();
```

- (c) Create a new `main()` function in the file `test-driver.c`. This main file should use the functions in `move.c` to initialize the towers data structure, print it out, and push and pop different values to it and printing it out at various points. Add rules to your Makefile to compile `test_driver.c` and link this with other object files to produce an executable. Run this program and check to make sure the output and debug if necessary before continuing.
- (d) Modify the `main()` in `main.c` and `make_move()` in `move.c` to use the new functions in `move.c` to produce the required output.

*Submission.* Please create a README file divided into three sections: NOTES TO GRADER, PROGRAM VALIDATION, and CERTIFICATION.

1. Under NOTES TO GRADER, add any notes, you wish the TA/instructor to consider while grading.  
*Include a list any errors that were in the Assignment 3 submission and have now been corrected.*  
In addition, 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.'
2. Under PROGRAM VALIDATION, please describe what you did to test your program or otherwise check that it is behaving correctly. (In particular: What inputs did you try? How did you determine if the outputs were correct? Did you try invalid inputs and make sure that your program doesn't crash?)
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. *Please also note whether you requested, received, and/or used a solution to Assignment 3 from the TA/instructor.*

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 the `assgn4` directory in the subversion repository is expected to include: `assignment4.pdf`, `README`, `Makefile`, `hanoi.h`, `main.c`, `hanoi.c`, `move.c`, `test-driver.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.

*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 there 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.