

0x19. C - Stacks, Queues - LIFO, FIFO

C

Group project

Algorithm

Data structure

By: Julien Barbier

Weight: 2

Project to be done in teams of 2 people (your team: Esubalew Kebede)

Project over - took place from Dec 13, 2022 6:00 AM to Dec 16, 2022 6:00 AM

An auto review will be launched at the deadline

In a nutshell...

• Contribution: 100.0%

• Auto QA review: 27.95/43 mandatory & 56.55/87 optional

Altogether: 107.25%
 Mandatory: 65.0%

Optional: 65.0%Contribution: 100.0%

Calculation: 100.0% * (65.0% + (65.0% * 65.0%)) == 107.25%



Resources

Read or watch:

- Google
- How do I use extern to share variables between source files in C?
- Stacks and Queues in C
- Stack operations



Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, without the help of Google:

General

- · What do LIFO and FIFO mean
- · What is a stack, and when to use it
- · What is a queue, and when to use it
- · What are the common implementations of stacks and queues
- · What are the most common use cases of stacks and queues
- What is the proper way to use global variables

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- · You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: vi , vim , emacs
- All your files will be compiled on Ubuntu 20.04 LTS using gcc, using the options -Wall -Werror -Wextra -pedantic std=c89
- · All your files should end with a new line
- A README.md file, at the root of the folder of the project is mandatory
- Your code should use the Betty style. It will be checked using betty-style.pl and betty-doc.pl
- You allowed to use a maximum of one global variable
- No more than 5 functions per file
- You are allowed to use the C standard library
- The prototypes of all your functions should be included in your header file called monty.h
- · Don't forget to push your header file
- · All your header files should be include guarded
- You are expected to do the tasks in the order shown in the project

GitHub

There should be one project repository per group. If you clone/fork/whatever a project repository with the same name before the second deadline, you risk a 0% score.

More Info

Data structures

Please use the following data structures for this project. Don't forget to include them in your header file.



```
/**
 * struct instruction_s - opcode and its function
 * @opcode: the opcode
 * @f: function to handle the opcode
 *
 * Description: opcode and its function
 * for stack, queues, LIFO, FIFO
 */
typedef struct instruction_s
{
    char *opcode;
    void (*f)(stack_t **stack, unsigned int line_number);
} instruction_t;
```

Compilation & Output

· Your code will be compiled this way:

```
$ gcc -Wall -Werror -Wextra -pedantic -std=c89 *.c -o monty
```

- Any output must be printed on stdout
- Any error message must be printed on stdern
 - Here is a link to a GitHub repository that could help you making sure your errors are printed on stdern

Tests

We strongly encourage you to work all together on a set of tests

The Monty language

Monty 0.98 is a scripting language that is first compiled into Monty byte codes (Just like Python). It relies on a unique stack, with specific instructions to manipulate it. The goal of this project is to create an interpreter for Monty ByteCodes files.

Monty byte code files

Files containing Monty byte codes usually have the mextension. Most of the industry uses this standard but it is not required by the specification of the language. There is not more than one instruction per line. There can be any number of spaces before or after the opcode and its argument:



Monty byte code files can contain blank lines (empty or made of spaces only, and any additional text after the opcode or its required argument is not taken into account:

```
julien@ubuntu:~/monty$ cat -e bytecodes/001.m
push 0 Push 0 onto the stack$
push 1 Push 1 onto the stack$
push 2$
 push 3$
                   pall
                            $
$
$
                            $
push 4$
    push 5
                        $
      push
$
pall This is the end of our program. Monty is awesome!$
julien@ubuntu:~/monty$
```

The monty program

- Usage: monty file
 - where file is the path to the file containing Monty byte code
- If the user does not give any file or more than one argument to your program, print the error message USAGE:
 monty file, followed by a new line, and exit with the status EXIT FAILURE
- If, for any reason, it's not possible to open the file, print the error message <a href="Error: Can't open file <file">Error: Can't open file <file, followed by a new line, and exit with the status EXIT FAILURE
 - where <file> is the name of the file
- - where is the line number where the instruction appears.
 - Line numbers always start at 1
- The monty program runs the bytecodes line by line and stop if either:
 - it executed properly every line of the file
 - o it finds an error in the file
 - an error occured
- If you can't malloc anymore, print the error message Error: malloc failed, followed by a new line, and exit with status EXXIT_FAILURE.
- You have to use malloc and free and are not allowed to use any other function from man malloc (realloc, calloc, ...)



Tasks

0. push, pall

Score: 65.0% (Checks completed: 100.0%)

Implement the push and pall opcodes.

The push opcode

The opcode push pushes an element to the stack.

- Usage: push <int>where <int> is an integer
- if <int> is not an integer or if there is no argument given to push, print the error message Lline_number>:
 usage: push integer, followed by a new line, and exit with the status EXIT_FAILURE
 - where is the line number in the file
- You won't have to deal with overflows. Use the atoi function

The pall opcode

The opcode pall prints all the values on the stack, starting from the top of the stack.

- Usage pall
- Format: see example
- If the stack is empty, don't print anything

```
julien@ubuntu:~/monty$ cat -e bytecodes/00.m
push 1$
push 2$
push 3$
pall$
julien@ubuntu:~/monty$ ./monty bytecodes/00.m
3
2
1
julien@ubuntu:~/monty$
```

Repo:

GitHub repository: monty

Done! Help Check your code Get a sandbox QA Review

1. pint

mandatory

Score: 65.0% (Checks completed: 100.0%)



The opcode pint prints the value at the top of the stack, followed by a new line.

- Usage: pint
- If the stack is empty, print the error message LLLLempty int, stack empty interest.

```
julien@ubuntu:~/monty$ cat bytecodes/06.m
push 1
pint
push 2
pint
push 3
pint
julien@ubuntu:~/monty$ ./monty bytecodes/06.m
1
2
3
julien@ubuntu:~/monty$
```

Repo:

GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review

2. pop

mandatory

Score: 65.0% (Checks completed: 100.0%)

Implement the pop opcode.

The pop opcode

The opcode pop removes the top element of the stack.

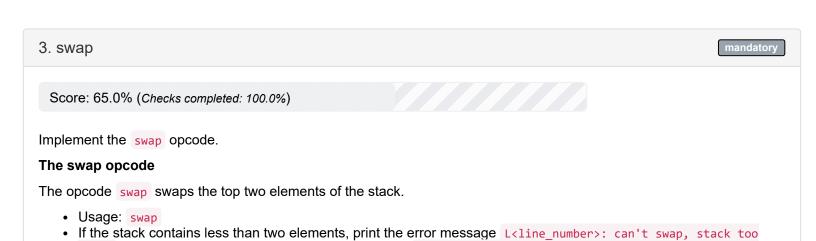
- Usage: pop
- If the stack is empty, print the error message LLLempty stack and exit with the status EXIT_FAILURE



```
push 2
 push 3
 pall
 pop
 pall
 pop
 pall
 pop
 pall
 julien@ubuntu:~/monty$ ./monty bytecodes/07.m
 2
 1
 2
 1
 1
 julien@ubuntu:~/monty$
Repo:

    GitHub repository: monty

                   Check your code
                                      Get a sandbox
                                                      QA Review
  Done!
           Help
```



short, followed by a new line, and exit with the status **EXIT_FAILURE**



```
push 2
 push 3
 pall
 swap
 pall
 julien@ubuntu:~/monty$ ./monty bytecodes/09.m
 3
 2
 1
 2
 3
 1
 julien@ubuntu:~/monty$
Repo:
   • GitHub repository: monty
                                     Get a sandbox
                                                     QA Review
  Done!
           Help
                   Check your code
```



Implement the add opcode.

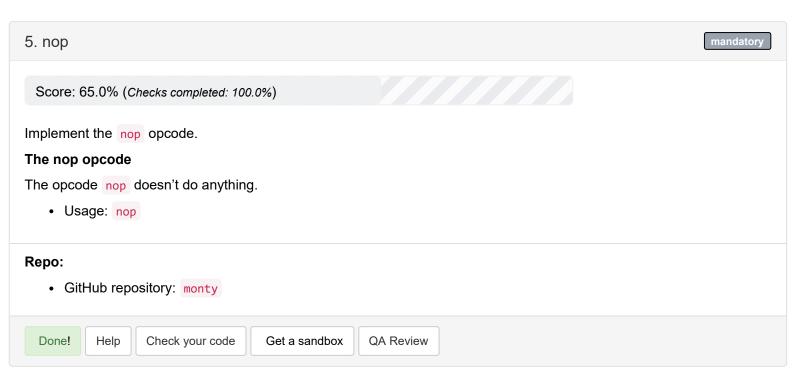
The add opcode

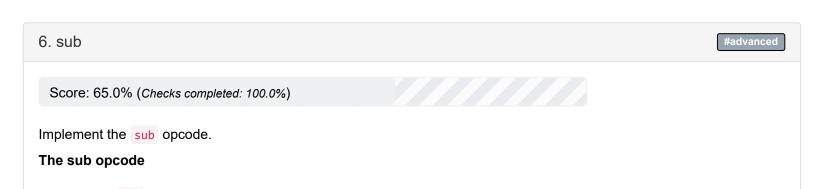
The opcode add adds the top two elements of the stack.

- Usage: add
- The result is stored in the second top element of the stack, and the top element is removed, so that at the end:
 - The top element of the stack contains the result
 - The stack is one element shorter

```
αl×
```

```
push 2
 push 3
 pall
 add
 pall
 julien@ubuntu:~/monty$ ./monty bytecodes/12.m
 3
 2
 1
 5
 1
 julien@ubuntu:~/monty$
Repo:
   • GitHub repository: monty
           Help
                   Check your code
                                     Get a sandbox
                                                     QA Review
  Done!
```







#advanced

- short, followed by a new line, and exit with the status EXIT FAILURE
- The result is stored in the second top element of the stack, and the top element is removed, so that at the end:
 - The top element of the stack contains the result
 - The stack is one element shorter

```
julien@ubuntu:~/monty$ cat bytecodes/19.m
push 1
push 2
push 10
push 3
sub
pall
julien@ubuntu:~/monty$ ./monty bytecodes/19.m
7
2
1
julien@ubuntu:~/monty$
```

Repo:

GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review

7. div

Score: 65.0% (Checks completed: 100.0%)

Implement the div opcode.

The div opcode

The opcode div divides the second top element of the stack by the top element of the stack.

- Usage: div
- The result is stored in the second top element of the stack, and the top element is removed, so that at the end:
 - The top element of the stack contains the result
 - The stack is one element shorter
- If the top element of the stack is 0, print the error message LLLdivision by zero, followed by a new line, and exit with the status EXIT FAILURE

Repo:

GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review



Score: 65.0% (Checks completed: 100.0%)

Implement the mul opcode.

The mul opcode

The opcode mul multiplies the second top element of the stack with the top element of the stack.

- Usage: mul
- The result is stored in the second top element of the stack, and the top element is removed, so that at the end:
 - The top element of the stack contains the result
 - The stack is one element shorter

Repo:

GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review

9. mod

#advanced

Score: 65.0% (Checks completed: 100.0%)

Implement the mod opcode.

The mod opcode

The opcode mod computes the rest of the division of the second top element of the stack by the top element of the stack.

- Usage: mod
- The result is stored in the second top element of the stack, and the top element is removed, so that at the end:
 - The top element of the stack contains the result
 - The stack is one element shorter
- If the top element of the stack is 0, print the error message LLLLdivision by zero, followed by a new line, and exit with the status EXIT_FAILURE

Repo:

GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review

10. comments

#advanced

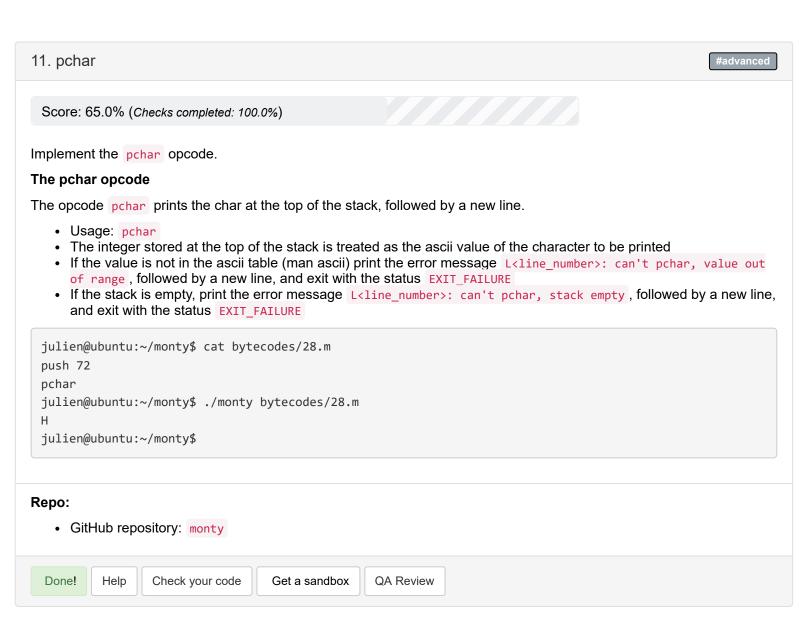




Repo:

• GitHub repository: monty

Done! Help Check your code Get a sandbox QA Review



12. pstr

Score: 65.0% (Checks completed: 100.0%)

Implement the pstr opcode.

The pstr opcode

The opcode pstr prints the string starting at the top of the stack, followed by a new line.



- the stack is over
- the value of the element is 0
- the value of the element is not in the ascii table
- If the stack is empty, print only a new line

```
julien@ubuntu:~/monty$ cat bytecodes/31.m
push 1
push 2
push 3
push 4
push 0
push 110
push 0
push 108
push 111
push 111
push 104
push 99
push 83
julien@ubuntu:~/monty$ ./monty bytecodes/31.m
School
julien@ubuntu:~/monty$
```

Repo:

GitHub repository: monty

Done! Help Check your code Get a sandbox QA Review

13. rotl #advanced

Score: 65.0% (Checks completed: 100.0%)

Implement the rot1 opcode.

The rotl opcode

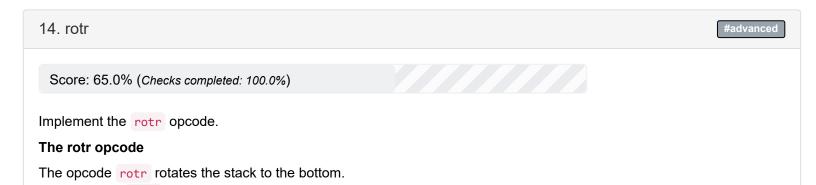
The opcode rot1 rotates the stack to the top.

- Usage: rot1
- The top element of the stack becomes the last one, and the second top element of the stack becomes the first one
- rot1 never fails

```
al×
```



```
push 2
 push 3
 push 4
 push 5
 push 6
 push 7
 push 8
 push 9
 push 0
 pall
 rotl
 pall
 julien@ubuntu:~/monty$ ./monty bytecodes/35.m
 9
 8
 7
 6
 5
 4
 3
 2
 1
 9
 8
 7
 6
 5
 4
 3
 2
 1
 julien@ubuntu:~/monty$
Repo:
   • GitHub repository: monty
  Done!
           Help
                   Check your code
                                     Get a sandbox
                                                     QA Review
```





Repo:

• GitHub repository: monty

Done!

Help

Check your code

Get a sandbox

QA Review

15. stack, queue

#advanced

Score: 65.0% (Checks completed: 100.0%)

Implement the stack and queue opcodes.

The stack opcode

The opcode stack sets the format of the data to a stack (LIFO). This is the default behavior of the program.

• Usage: stack

The queue opcode

The opcode queue sets the format of the data to a queue (FIFO).

• Usage: queue

When switching mode:

- The top of the stack becomes the front of the queue
- The front of the queue becomes the top of the stack

```
al×
```



```
push 1
 push 2
 push 3
 pall
 stack
 push 4
 push 5
 push 6
 pall
 add
 pall
 queue
 push 11111
 add
 pall
 julien@ubuntu:~/monty$ ./monty bytecodes/47.m
 2
 3
 6
 5
 4
 1
 2
 3
 11
 4
 1
 2
 3
 15
 1
 2
 3
 11111
 julien@ubuntu:~/monty$
Repo:
   • GitHub repository: monty
```

Done! Help Check your code Get a sandbox QA Review

16. Brainf*ck

#advanced

Score: 65.0% (Checks completed: 100.0%)

Write a Brainf*ck script that prints <a>School, followed by a new line.



```
julien@ubuntu:~/monty/bf$ bf 1000-school.bf
 School
 julien@ubuntu:~/monty/bf$
Repo:

    GitHub repository: monty

    · Directory: bf
    • File: 1000-school.bf
                                      Get a sandbox
                                                       QA Review
  Done!
           Help
                   Check your code
17. Add two digits
                                                                                                             #advanced
 Score: 65.0% (Checks completed: 100.0%)
Add two digits given by the user.
    · Read the two digits from stdin, add them, and print the result
    • The total of the two digits with be one digit-long (<10)
 julien@ubuntu:~/monty/bf$ bf ./1001-add.bf
 81
 9julien@ubuntu:~/monty/bf$
Repo:

    GitHub repository: monty

    · Directory: bf
    • File: 1001-add.bf
  Done!
           Help
                   Check your code
                                      Get a sandbox
                                                       QA Review
```

18. Multiplication

#advanced

Score: 65.0% (Checks completed: 100.0%)

Multiply two digits given by the user.

- · Read the two digits from stdin, multiply them, and print the result
- The result of the multiplication will be one digit-long (<10)



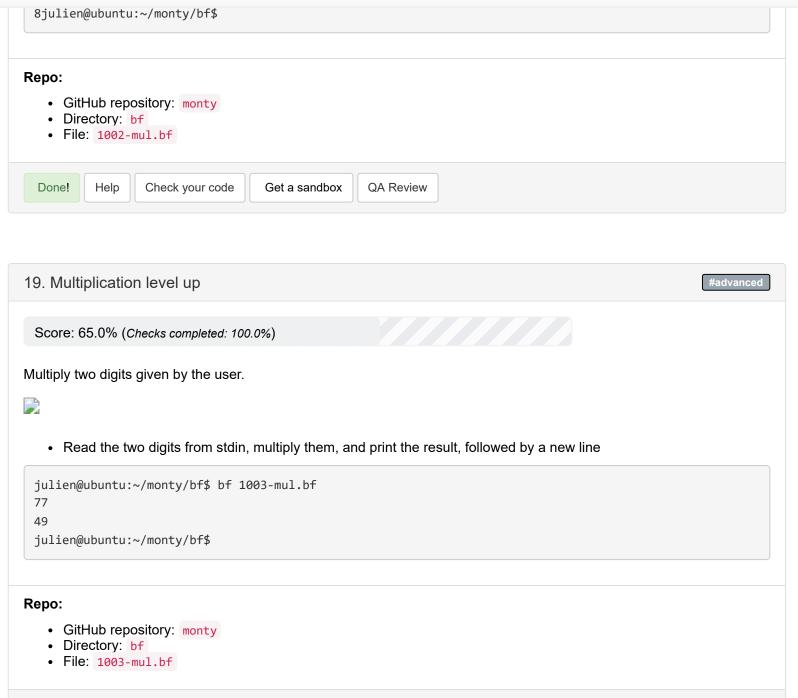
Help

Done!

Check your code

Get a sandbox





QA Review

Copyright © 2022 ALX, All rights reserved.