Project 2: Question 1

Growth Rates With Selected Inputs

Below is a illustration of the growth rates from the fuctions provided for question 1. Using "Big-Oh" we are able to order the growth rates.

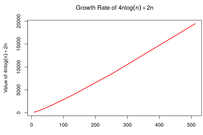
```
library(knitr)
library(tableExtra)
n <- c(n,2,4,8,16,256,512)
f1 <- 4*n*log2(n)+2*n
f2 <- 2^10
f3 <- 2*log2(n)
f4 <- 3*n + 100*log2(n)
f5 <- 4*n
f6 <- 2^n
f7 <- n^2 + 10*n
f8 <- n^3
f9 <- n*log2(n)
data <- data.frame(n,f1,f2,f3,f4,f5,f6,f7,f8,f9)
data <- round(data)
names <- c("n","4nlog(n) + 2n","$2^{10}$","2log(n)","3n + 100log(n)","4n","$2^n$","$n^2 + 10n$","$n^3$",
"nlogn")
data %>%
  kable(col.names = names) %>%
  kable_styling()
```

n	$4nlog(n) + 2n$	2^{10}	$2log(n)$	$3n + 100log(n)$	$4n$	2^n	$n^2 + 10n$	n^3	nlogn
8	112	1024	6	324	32	2.560000e+02	144	512	24
16	288	1024	8	448	64	6.553600e+04	416	4096	64
32	704	1024	10	596	128	4.294967e+09	1344	32768	160
64	1664	1024	12	792	256	1.844674e+19	4736	262144	384
128	3840	1024	14	1084	512	3.402824e+38	17664	2097152	896
256	8704	1024	16	1568	1024	1.157921e+77	68096	16777216	2048
512	19456	1024	18	2436	2048	1.340781e+154	267264	134217728	4608

Growth Rates

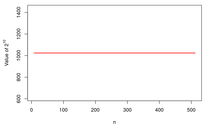
$4nlog(n) + 2n$

```
plot(data$n,data$f1,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ", 4*n*log(n)+2*
n), main = expression(paste("Growth Rate of ",4*n*log(n)+2*n)))
```



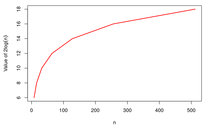
2^{10}

```
plot(data$n,data$f2,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ", 2^10)), main =
 expression(paste("Growth Rate of ",2^10)))
```



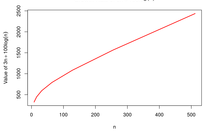
$2log(n)$

```
plot(data$n,data$f3,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ", 2*log(n)), mai
n = expression(paste("Growth Rate of ",2*log(n))))
```



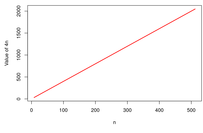
$3n + 100log(n)$

```
plot(data$n,data$f4,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ",3*n+100*log
(n)), main = expression(paste("Growth Rate of ",3*n+100*log(n))))
```



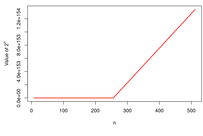
$4n$

```
plot(data$n,data$f5,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ", 4*n), main = e
xpression(paste("Growth Rate of ",4*n)))
```



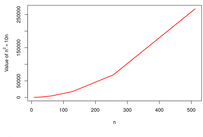
2^n

```
plot(data$n,data$f6,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ", 2^n), main = e
xpression(paste("Growth Rate of ",2^n)))
```



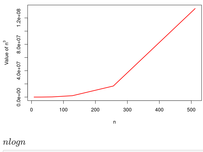
$n^2 + 10n$

```
plot(data$n,data$f7,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ",n^2+10*n), mai
n = expression(paste("Growth Rate of ",n^2+10*n)))
```



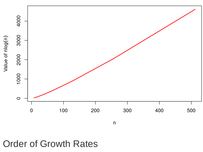
n^3

```
plot(data$n,data$f8,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ",n^3)), main = e
xpression(paste("Growth Rate of ",n^3)))
```



$nlogn$

```
plot(data$n,data$f9,type="l", col="red", lwd=2, xlab = "n", ylab = expression(paste("Value of ",n*log(n)), mai
n = expression(paste("Growth Rate of ",n*log(n))))
```



Order of Growth Rates

1. 2^n
   - 2^n is $O(2^n)$, $2^n \leq c^n$, where $c = 2$, when $n \geq n_0 = 1$
2. n^3
   - n^3 is $O(n^3)$, $n^3 \leq cn^3$, where $c = 1$, when $n \geq n_0 = 1$
3. $n^2 + 10n$
   - $n^2 + 10n$ is $O(n^2)$, $n^2 + 10n \leq cn^2$, where $c = 11$, when $n \geq n_0 = 1$
4. $4nlogn + 2n$
   - $4nlogn + 2n$ is $O(nlogn)$, $4nlogn + 2n \leq cnlogn$, where $c = 6$, when $n \geq n_0 = 2$
5. $nlogn$
   - $nlogn$ is $O(nlogn)$, $nlogn \leq cnlogn$, where $c = 1$, when $n \geq n_0 = 2$
6. $4n$
   - $4n$ is $O(n)$, $4n \leq cn$, where $c = 4$, when $n \geq n_0 = 1$
7. $3n + 100log n$
   - $3n + 100log n$ is $O(n)$, $3n + 100log n \leq cn$, where $c = 103$, when $n \geq n_0 = 1$
8. $2log n$
   - $2log n$ is $O(logn)$, $2log n \leq clogn$, where $c = 2$, when $n \geq n_0 = 2$
9. 2^{10}
   - 2^{10} is $O(1)$

Comparing Growth Rates

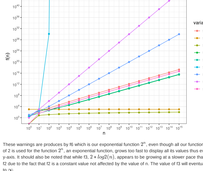
```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(reshape2)
library(scales)
theme_set(theme_bw())
n <- 2^(0:10)
n2 <- 10^seq(1,64,4)

f1 <- 4*n*log2(n)+2
f2 <- 2^10
f3 <- 2*log2(n)
f4 <- 3*n + 100*log2(n)
f5 <- 4*n
f6 <- 2^n
f7 <- n^2 + 10*n
f8 <- n^3
f9 <- n*log2(n)

data <- data.frame(n,f1,f2,f3,f4,f5,f6,f7,f8,f9)
data <- round(data)
names <- c("n2","4nlog(n)+2","$2^{10}$","2log(n)","3n+100log(n)","4n","$2^n$","$n^2 + 10n$","$n^3$","n
logn")
ggplot(melt(data = data, "n"), aes(n, value, colour=variable)) +
  geom_line() + geom_point() +
  scale_x_continuous(expression(n),breaks = n,trans = 'log10',labels=trans_format('log10',math_format(10^.x))) +
  scale_y_continuous(expression(f(n)),breaks = n2,trans = 'log10',labels = trans_format('log10',math_format(10
^.x))) +
  coord_cartesian(ylim = c(0.5^10,1^11))
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

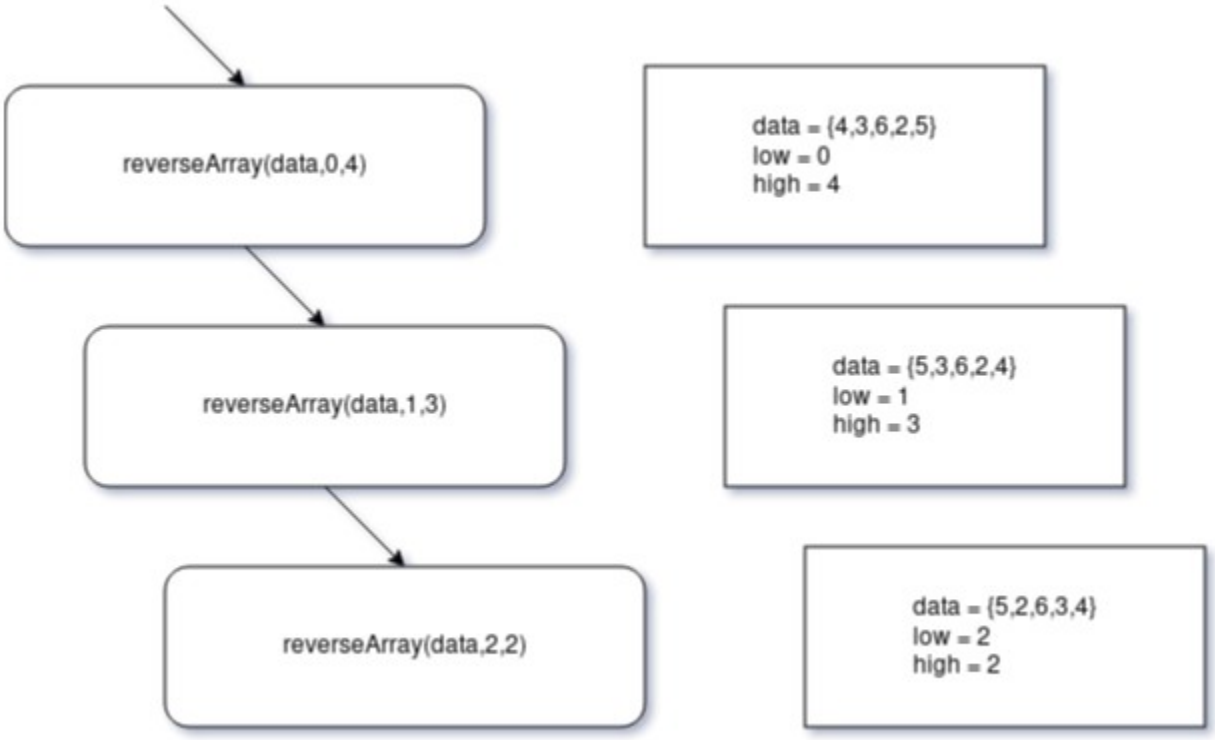
```
## Warning: Transformation introduced infinite values in continuous y-axis
```



These warnings are produces by fill which is our exponential function 2^n, even though all our functions are plotted on a log-log chart and a base of 2 is used for the function 2^n, an exponential function, grows too fast to display all its values thus eventually leading to an infinite value in the y-axis. It should also be noted that while $0.2 * log(0.5)$, appears to be growing at a slower pace than 2^n, 2^n IS is actually growing faster than f2 due to the fact that f2 is a constant value not affected by the value of n. The value of f2 will eventually surpass the value of f2 as it gets closer to ∞

Question 2:

Draw the recursion trace (refer to an example in Figure 5.1) for the execution of method reverseArray(data, 0, 4) (Code Fragment 5.7) on array data = {4, 3, 6, 2, 5}.



Question 3

Using linked-list Stacks, write a JAVA program that allows the user to enter a mathematical expression to check its validation. Implement the Parenthesis Matching Algorithm discussed in the lectures using linked-list Stack to be able to check whether a user's mathematical expression is valid.

```
/usr/lib/jvm/java-8-openjdk/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=39289:/usr/share/idea/bin
Enter a mathematical expression:
(3n+2)(2n+5))
Nothing to match ")".

Process finished with exit code 0
```

```
/usr/lib/jvm/java-8-openjdk/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=41859
Enter a mathematical expression:
(2n+4)(3m+5)[6n]
Everything matched!
```

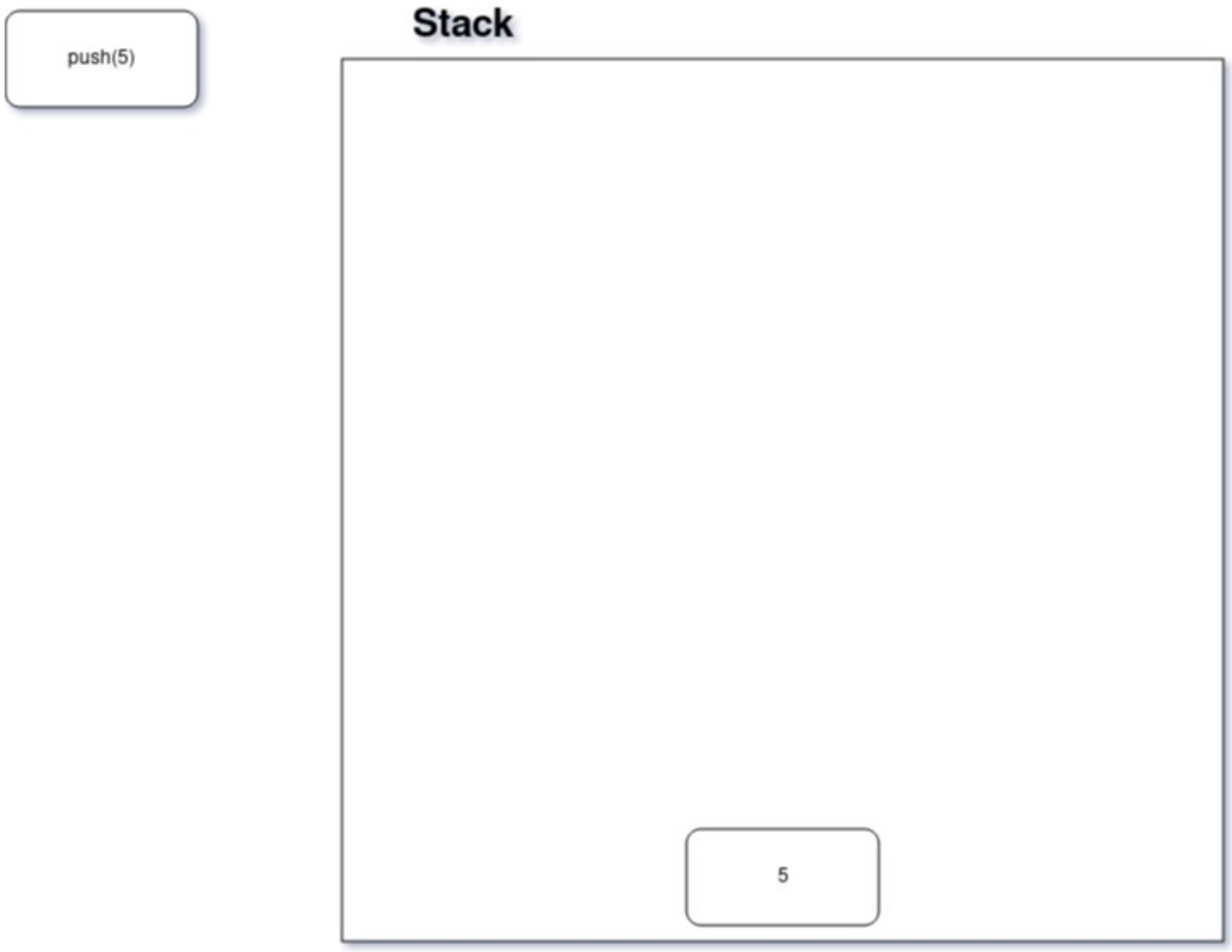
Parenthesis Matching Algorithm:

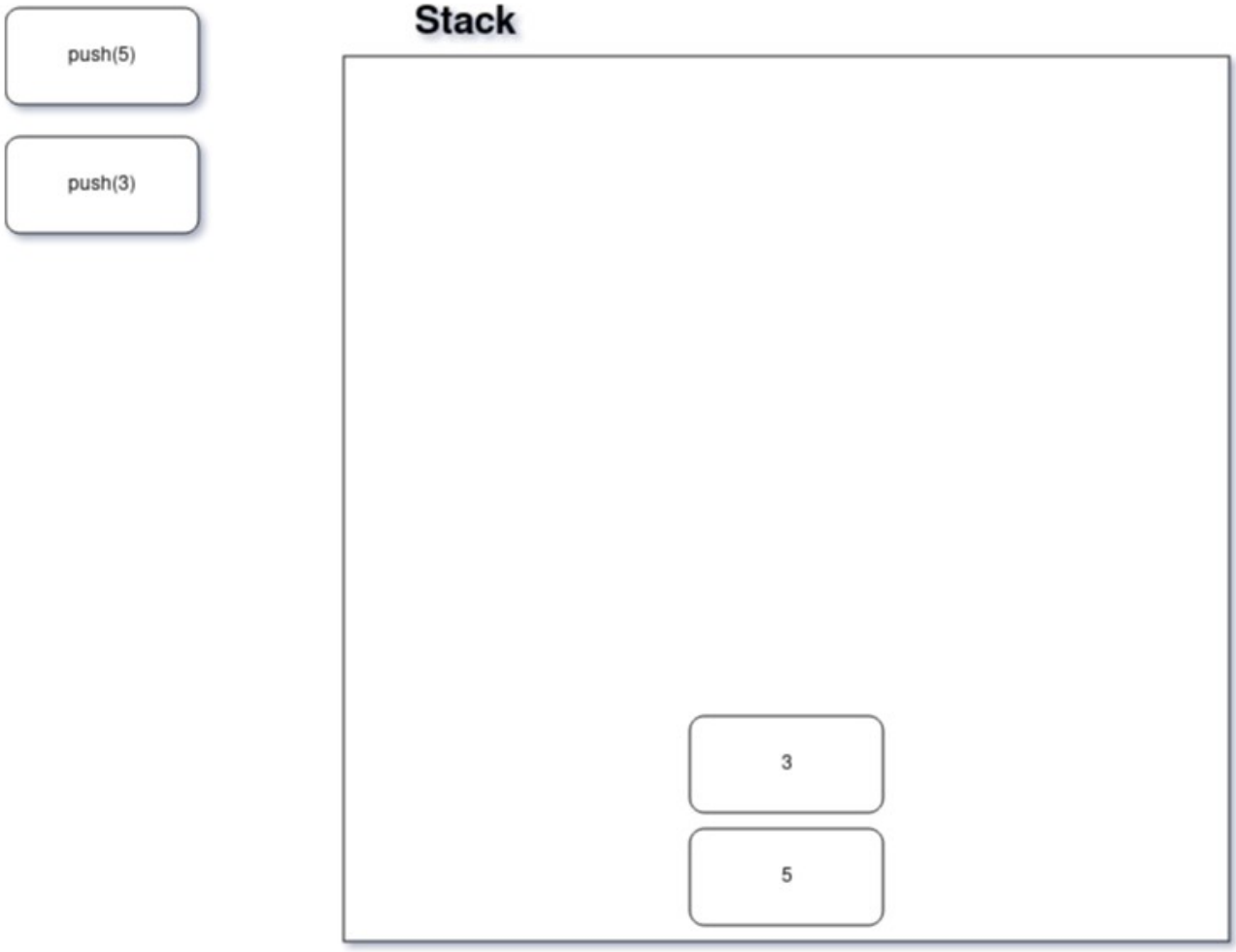
```
private boolean checkParenMatch(String x) {
    LinkedList<Node<String>> stack = new LinkedList<>();
    String[] expression = x.split( s: "");
    for (int i = 0; i < expression.length; i++) {
        if (expression[i].equals("(") || expression[i].equals("{") || expression[i].equals("[")) {
            Node<String> object = new Node<>(expression[i], p: null, n: null);
            stack.push(object);
        } else if (expression[i].equals(")") || expression[i].equals("}") || expression[i].equals("]")) {
            if (stack.isEmpty()) {
                System.out.println("Nothing to match \"" + expression[i] + "\".");
                return false;
            } else {
                if (stack.last().getElement().equals("(") && expression[i].equals(")")) {
                    stack.pop();
                } else if (stack.last().getElement().equals("{") && expression[i].equals("}")) {
                    stack.pop();
                } else if (stack.last().getElement().equals("[") && expression[i].equals("]")) {
                    stack.pop();
                }
            }
        }

    }
    if (stack.isEmpty()) {
        System.out.println("Everything matched!");
        return true;
    } else {
        System.out.println("Not everything matched.");
        return false;
    }
}
```

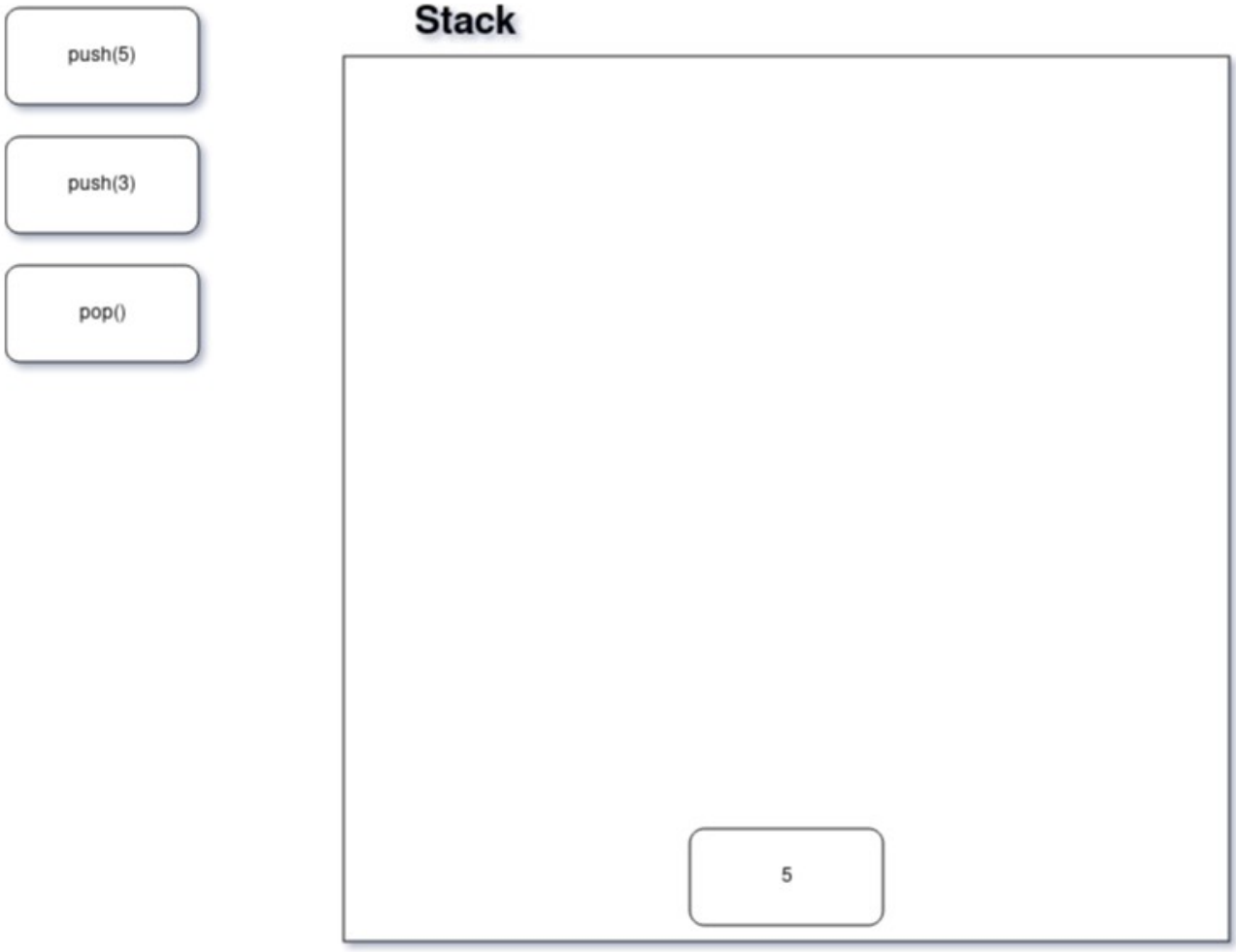
Question 4A

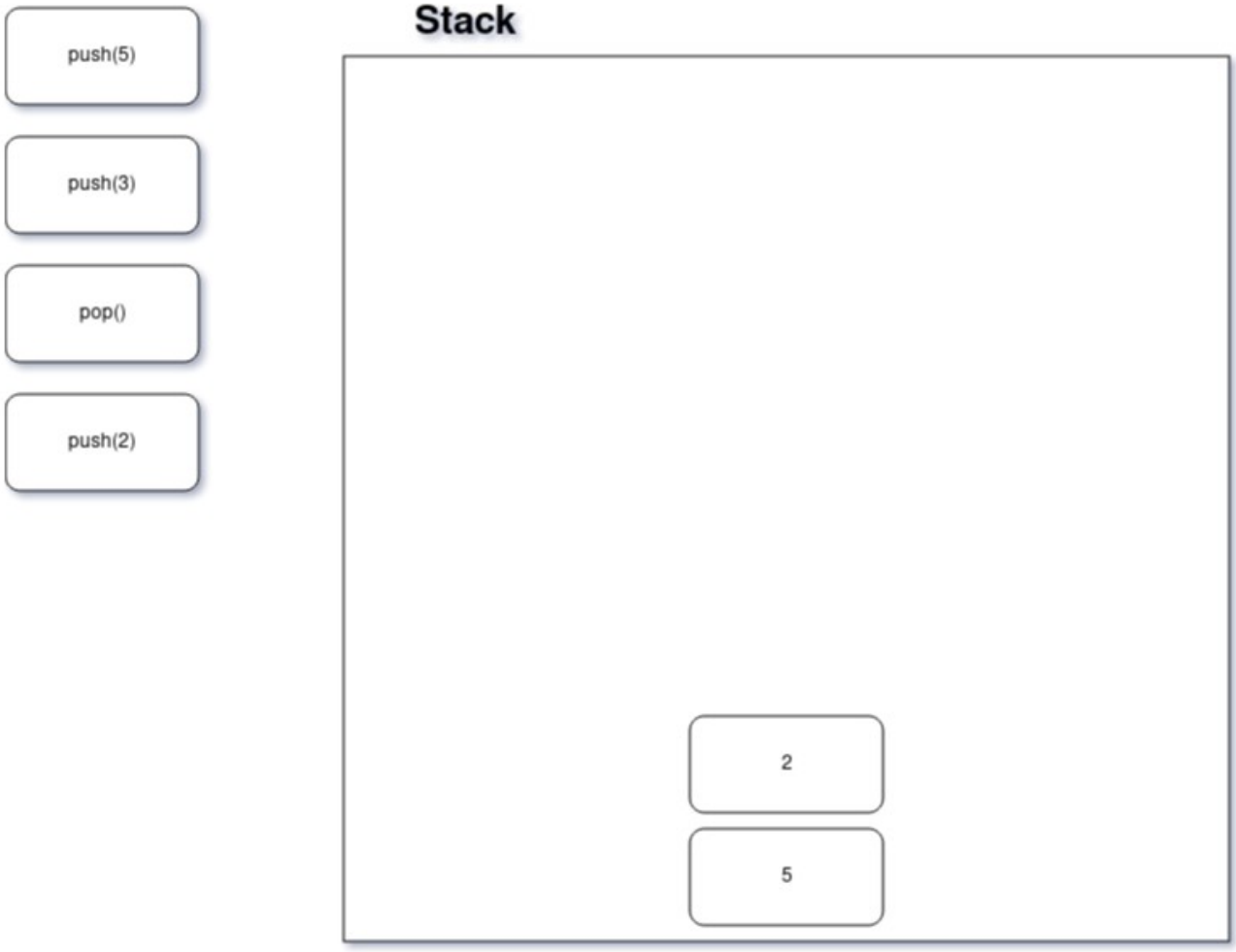
A. Illustrate the current state of the Stack after each of the following stack operation:

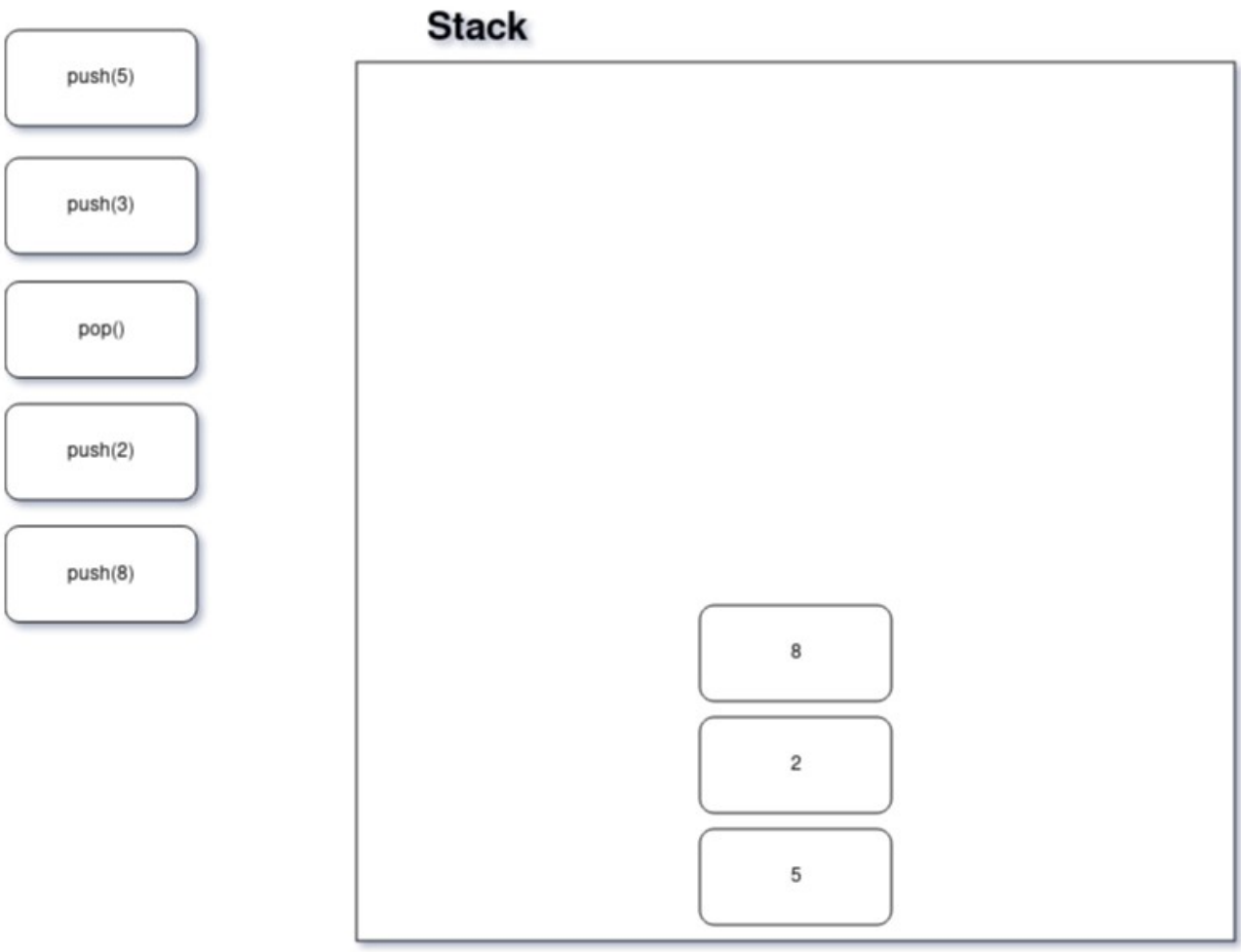
push(5), push(3), pop( ), push(2), push(8), pop( ), pop( ), push(9), push(1), pop( ), push(7), push(6), pop( ), pop( ), push(4), pop( ), pop( ).

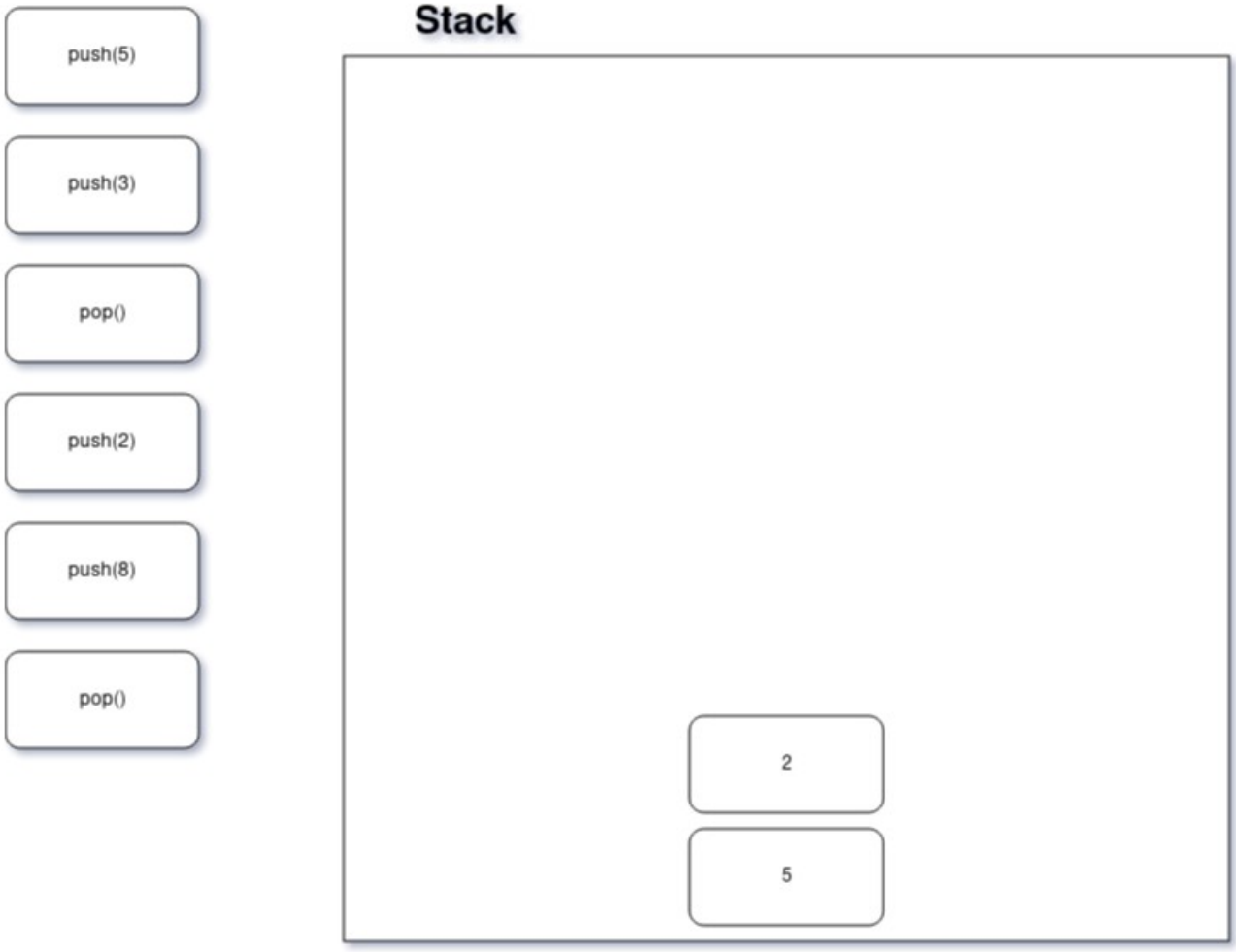


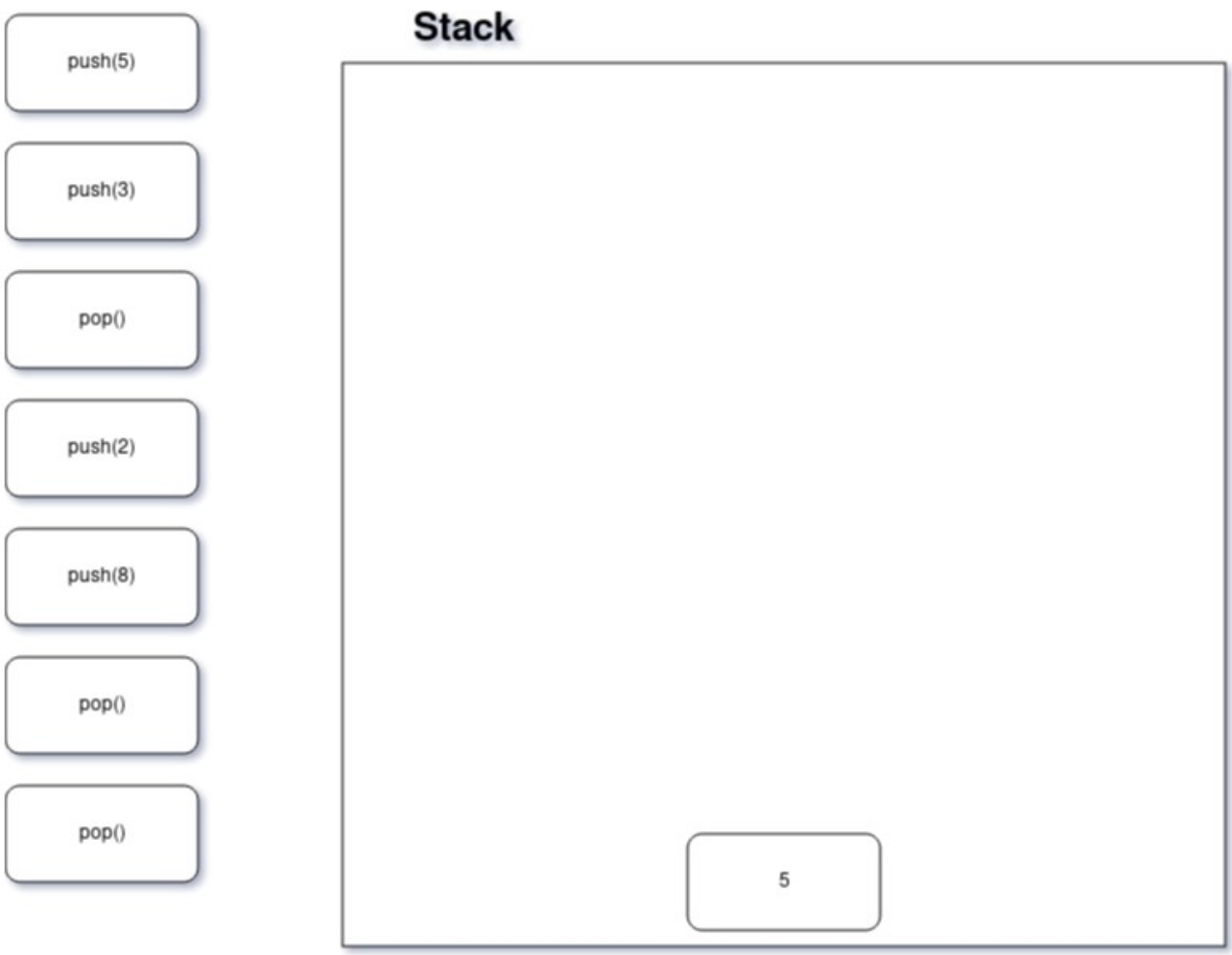

push(5)
push(3)
Stack
3
5

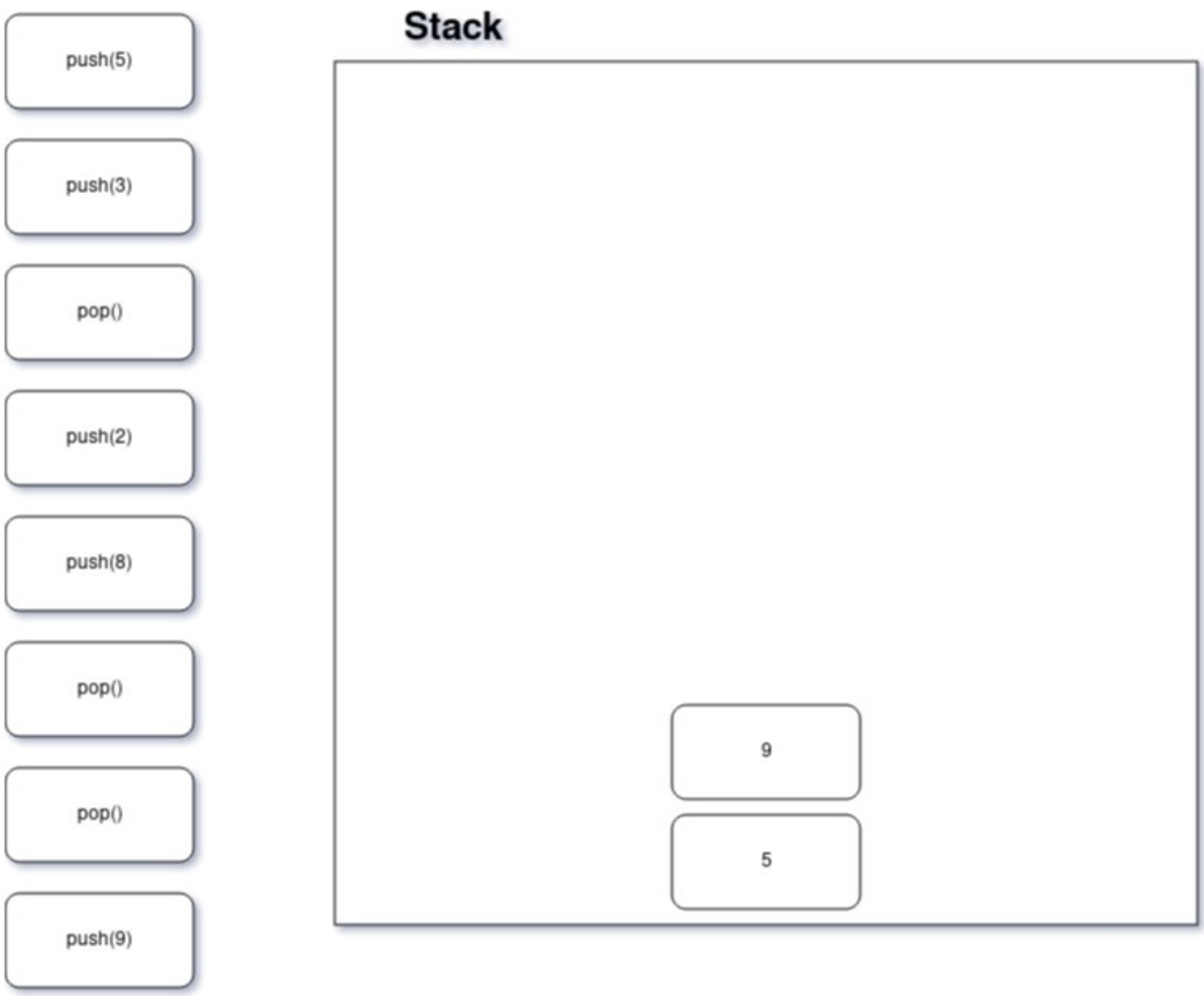

push(5)
push(3)
pop()
Stack
5

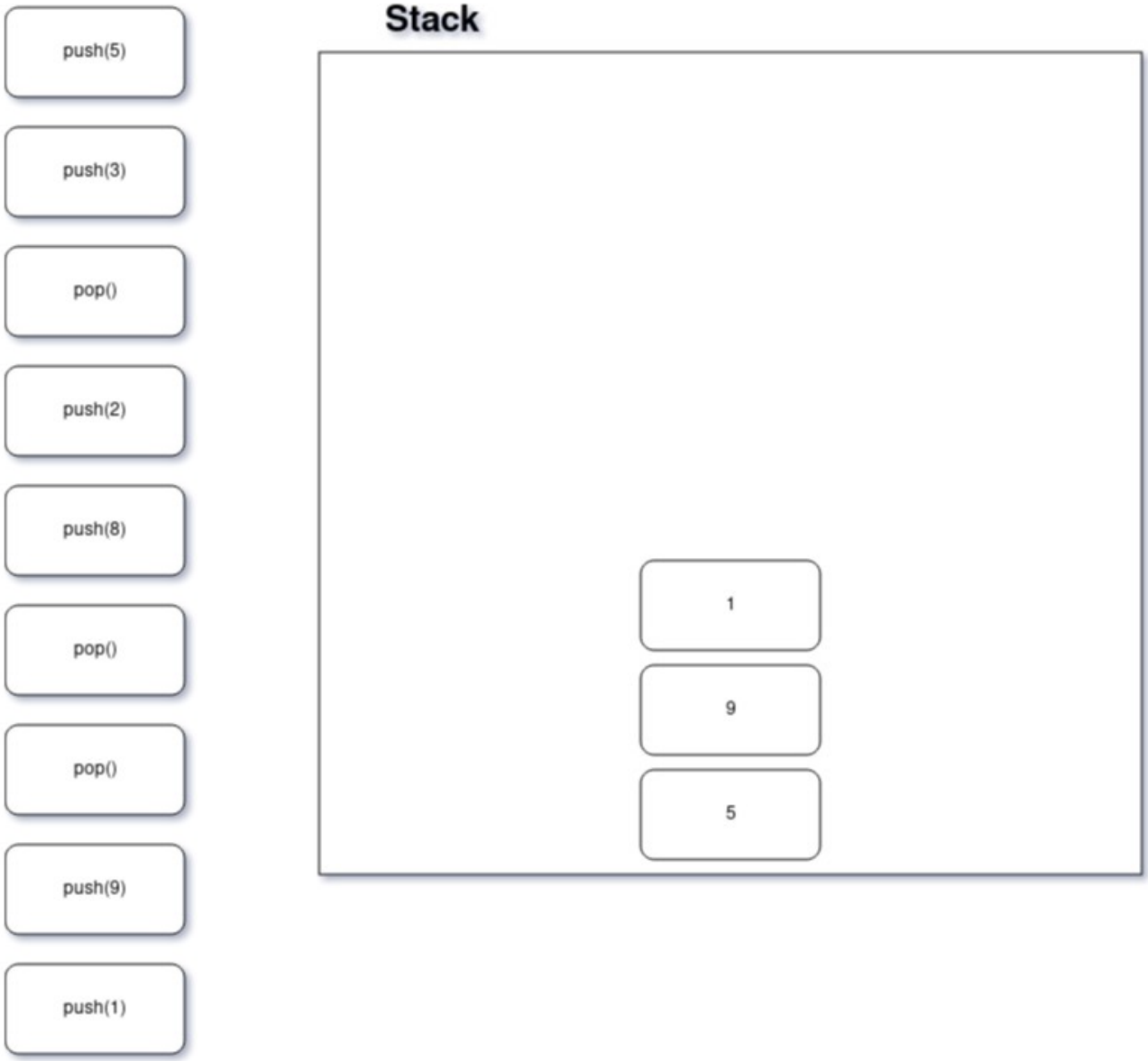

push(5)
push(3)
pop()
push(2)
Stack
2
5

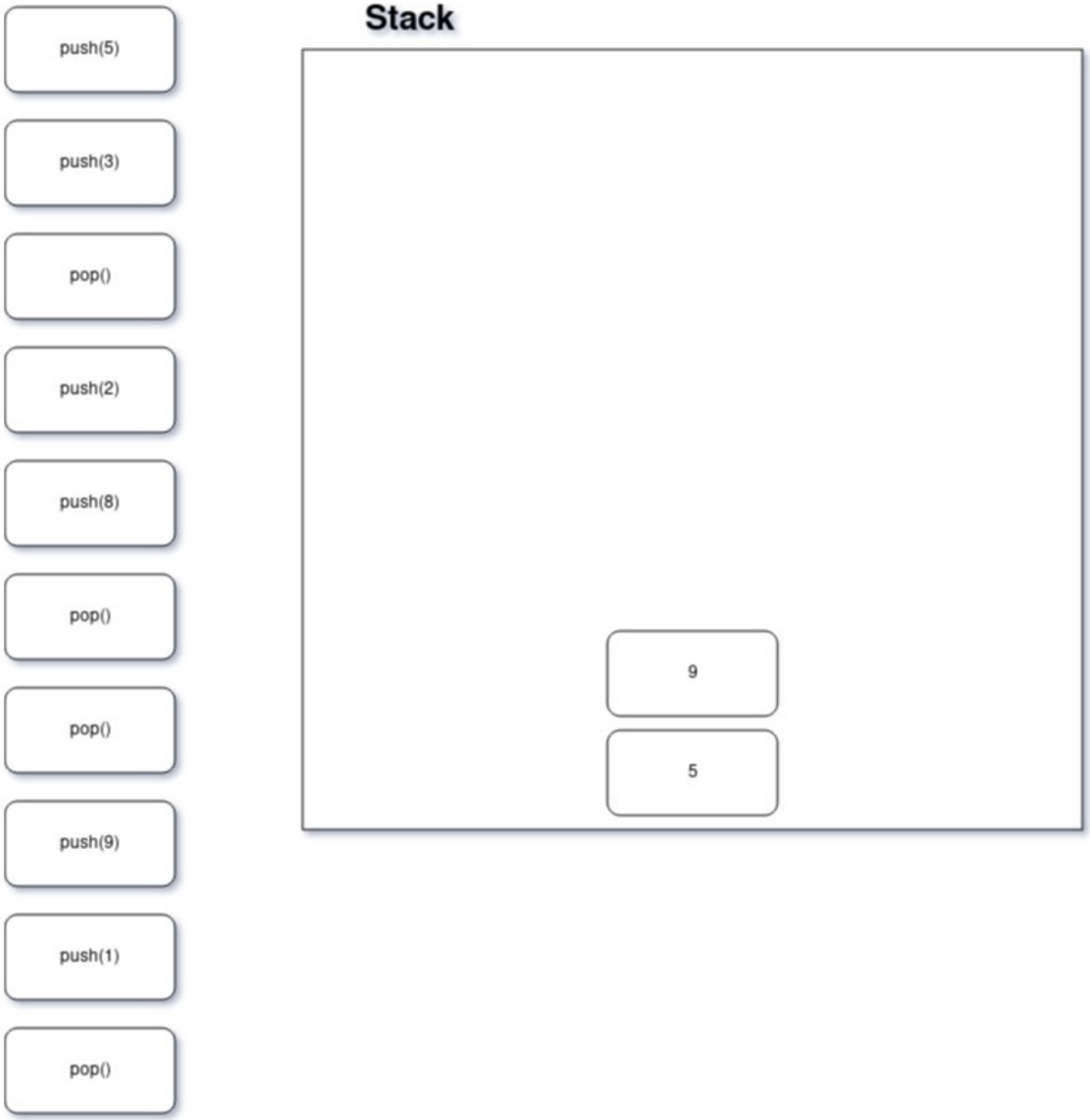

push(5)
push(3)
pop()
push(2)
push(8)
Stack
8
2
5

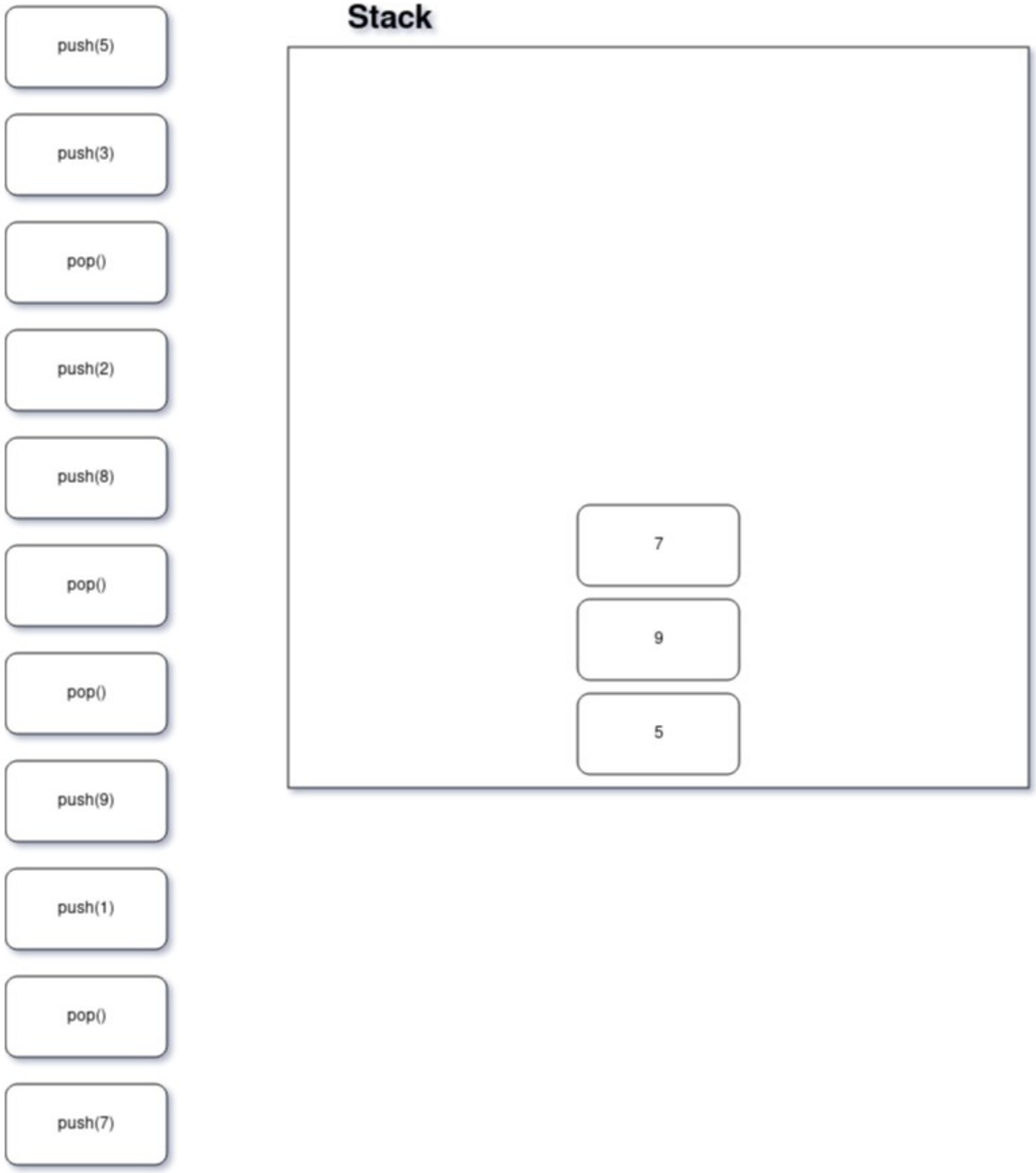

push(5)
push(3)
pop()
push(2)
push(8)
pop()
Stack
2
5

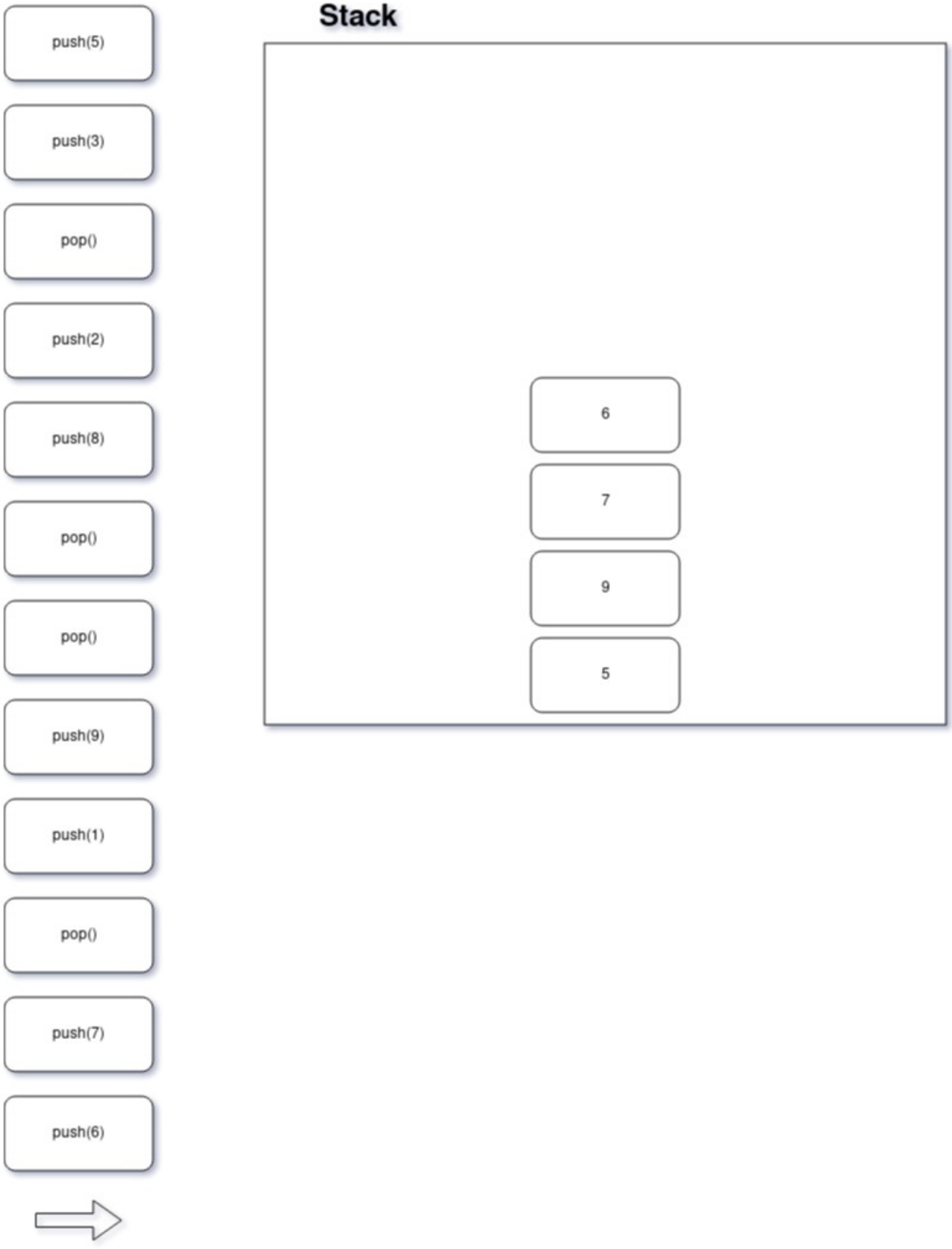

push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
Stack
5


push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
push(9)
Stack
9
5

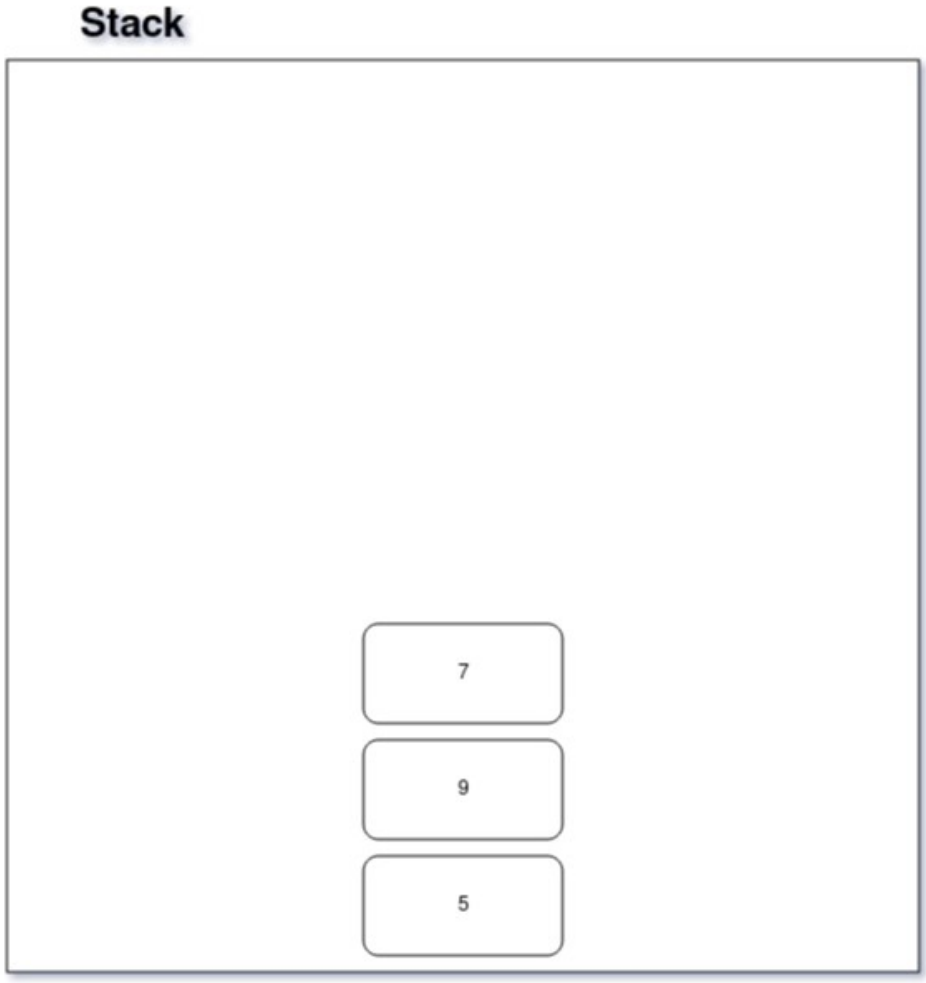

push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
push(9)
push(1)
Stack
1
9
5

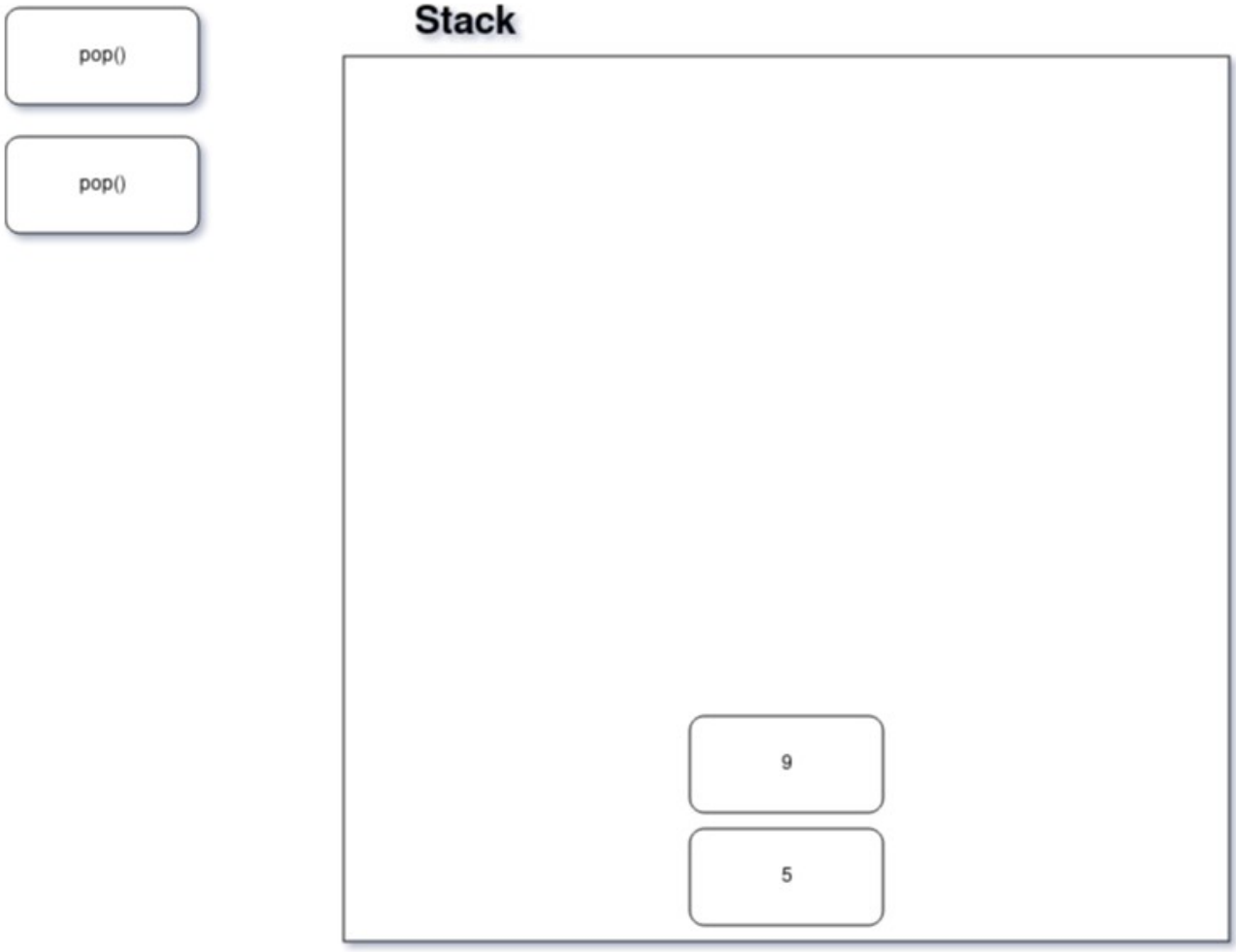

push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
push(9)
push(1)
pop()
Stack
9
5


push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
push(9)
push(1)
pop()
push(7)
Stack
7
9
5


push(5)
push(3)
pop()
push(2)
push(8)
pop()
pop()
push(9)
push(1)
pop()
push(7)
push(6)
Stack
6
7
9
5

pop()

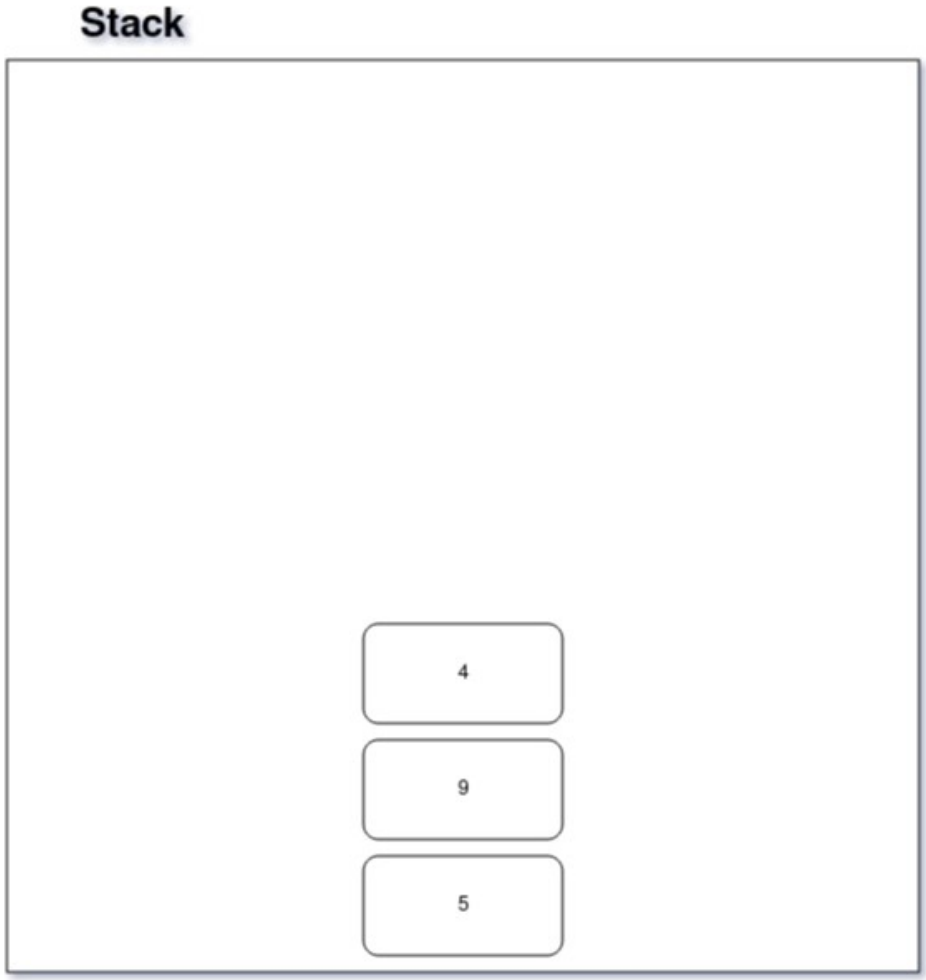


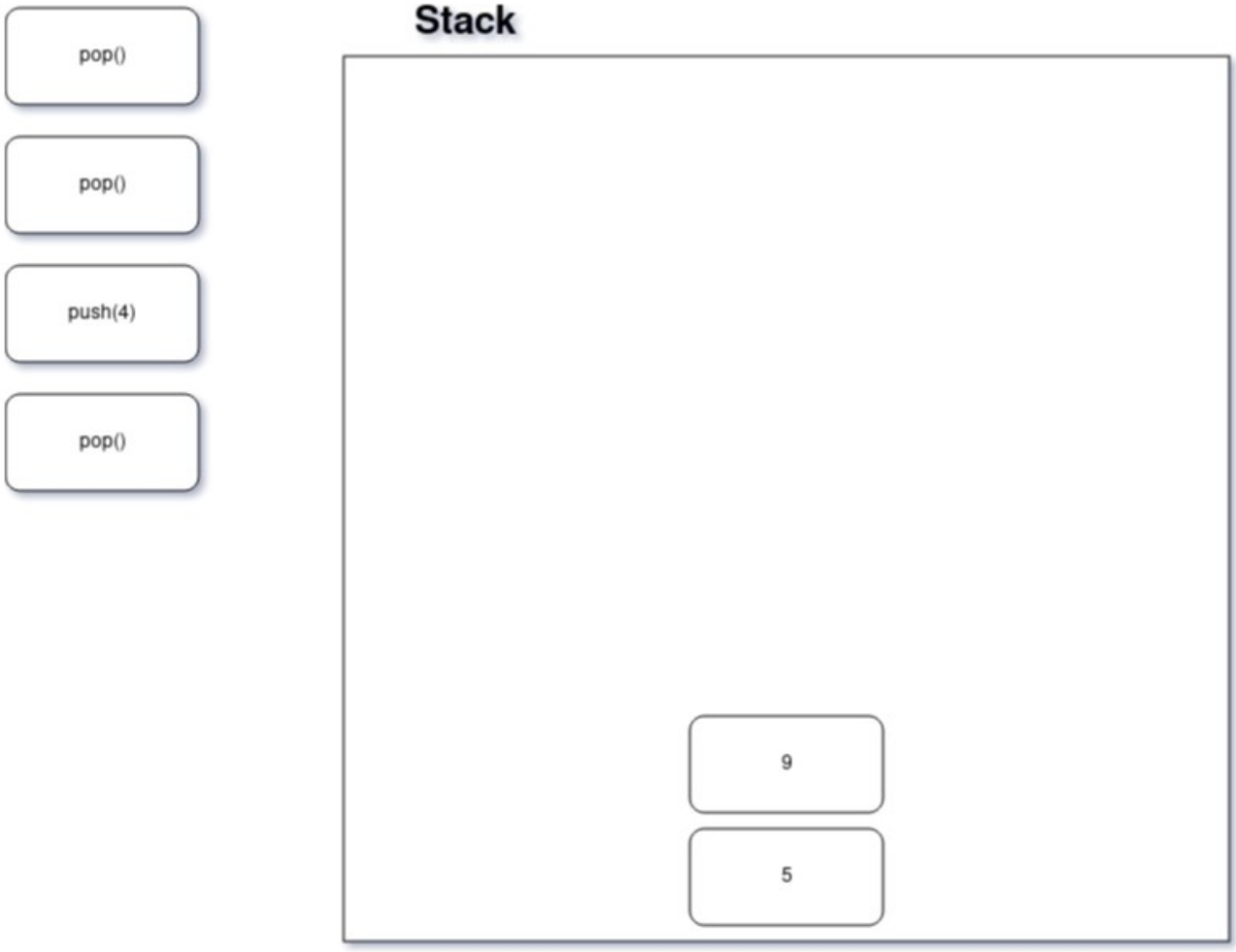

pop()
pop()
Stack
9
5

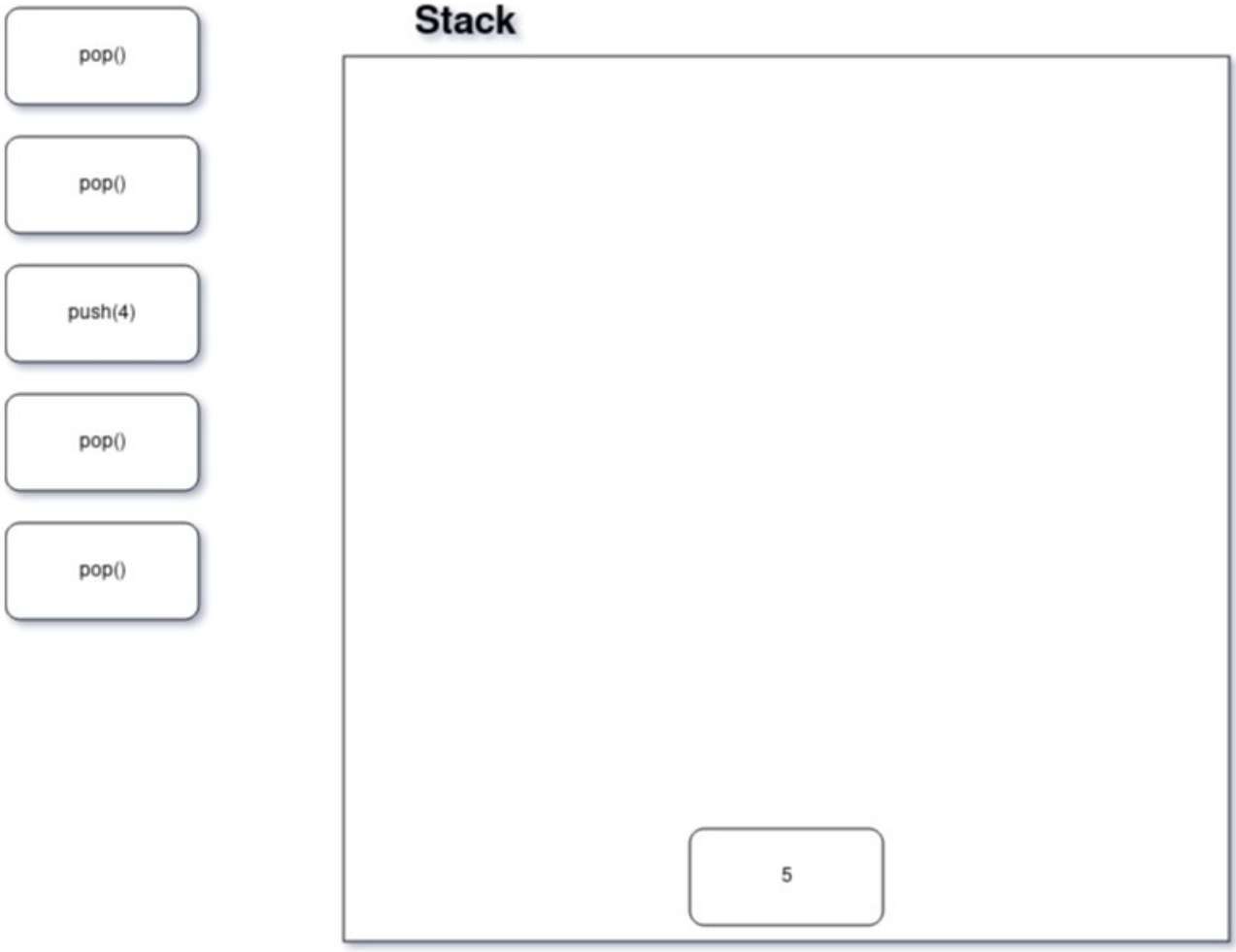
pop()

pop()

push(4)




pop()
pop()
push(4)
pop()
Stack
9
5


pop()
pop()
push(4)
pop()
pop()
Stack
5

Question 4b

B. Illustrate the current state of the Queue after each of the following queue operation:

enqueue(5), enqueue(3), dequeue( ), enqueue(2), enqueue(8), dequeue( ), dequeue( ), enqueue(9), enqueue(1), dequeue( ), enqueue(7), enqueue(6), dequeue( ), dequeue( ), enqueue(4), dequeue( ), dequeue( ).

enqueue(5)

enqueue(3)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(5)

enqueue(1)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(6)

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(6)

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(6)

dequeue( )

dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(6)

dequeue( )

dequeue( )

enqueue(4)

enqueue(5)	enqueue(1)
enqueue(3)	dequeue( )
dequeue( )	enqueue(7)
enqueue(2)	enqueue(6)
enqueue(8)	dequeue( )
dequeue( )	dequeue( )
dequeue( )	enqueue(4)
enqueue(9)	dequeue( )

enqueue(5)

enqueue(3)

dequeue( )

enqueue(2)

enqueue(8)

dequeue( )

dequeue( )

enqueue(9)

enqueue(1)

dequeue( )

enqueue(7)

enqueue(6)

dequeue( )

dequeue( )

enqueue(4)

dequeue( )

dequeue( )

Youtube Videos

https://youtu.be/9mZDUEgpXng

https://youtu.be/o2ZNYk0uaI0