STACKS AND QUEUES

SPECIFICATION

```
spec STACK[CHAR]
      genre stack, char
      operations
            find: stack char -> bool
            equal: stack stack -> bool
            findAll: stack stack -> bool
            count_vowels: stack -> int
            only_vowels: stack -> stack

endspec
```

IMPLEMENTATION

```
class Stack
      public bool find(stack,char)
      public bool equal(stack,stack)
      public bool find_all (stack,stack)
      public int count_vowels(stack)
      public stack only_vowels(stack)
endclass
```

Isabel Martínez Gómez 2E
Data Structures GII

```
public bool Stack::find_all(s1:stack,s2:stack)
      auxs2:stack
      c1,c2:char
      auxs2=s2
      if (s1.empty())
            return true
      endif
      else:
            c1=s1.pop()
            c2=auxs2.pop()
            while (c1!=c2)
                  if (auxs2.empty())
                        return false
                  endif
                  else
                        return true
                  endelse
            endwhile
      endelse
      return find_all(s1,s2)
endmethod
```

$O(n^2)$

The running time of "find_all" method is $O(n^2)$ because it's a recursive method in which inside there is a while loop that will executes n times and this method is going to be call n times.

```
public int Stack::count_vowels(s:stack)
      c:char
      auxs:stack
      auxs=s
      char vowels[5]={‘a’,’e’,’i’,’o’,’u’}
      if (auxs.empty())
            return 0
      endif
      else
            c=auxs.pop()
            for (int i=0;i<vowels.length();i++)
                  if (c==vowels[i])
                        return 1+count_vowels(auxs)
                  endif
                  else
                        return count_vowels(auxs)
                  endelse
            endfor
      endelse
endmethod
```

O(n)

The running time of “count_vowels” method is O(n) because it’s a recursive method which is going to be executed n times.

```
public stack Stack::only_vowels(s:stack)
      c:char
      vowelsStack,auxs:stack
      char vowels[5]={‘a’,’e’,’i’,’o’,’u’}
      auxs=s
      if (auxs.empty())
            return vowelsStack
      endif
      else
            c=auxs.pop()
            for (int i=0;i<vowels.length();i++)
                  if (c==vowels[i])
                        vowelsStack.push(c)
                  endif
            endfor
      endelse
      return only_vowels(auxs)
endmethod
```

O(n)

The running time of “count_vowels” method is O(n) because it’s a recursive method which is going to be executed n times.

Best possible running time of operations 1 and 2:

Operation 1

The best possible running time of “find” method is O(1) if we do it in an iterative way. We only need conditional statements to check if a character is in the stack.

The best possible running time of “find” method is O(n) if we do it in a recursive way, this is because the method will be executed n times so that we can look for the characters inside the stack and compare it with the introduced character.

Operation 2

The same that in operation 1 is going to be applied to operation 2.

The best possible running time of “equal” method is O(1) if we do it in an iterative way. We only need conditional statements to check if both characters in the stack are equal.

The best possible running time of “equal” method is O(n) if we do it in a recursive way, this is because the method will be executed n times so that we can look if the character pop of both stacks are equal or not.