ECS 140A Homework 2 – Problem 1

1 Python

Step 1: Algorithm/Pseudocode

```
class Stack:
    __init__(): # initializer
        # create empty array

    push(val):
        # append val to array

    pop():
        # use list.pop() function to remove most recently added item to array
        # return item removed

    peek():
        # access and return last item in array
```

Step 2: Actual Code

```
class Stack:
    def __init__(self):
        self.nums = []

    def push(self, val):
        self.nums.append(val)

    def pop(self):
        return self.nums.pop()

    def peek(self):
        return self.nums[-1]
```

Step 3: Working Code

There were no syntax errors, so the initial working code was the same as the previous step.

Step 4: Debug Process

All test cases (found in step 6) behaved as expected.

```
(base) Annas-MacBook-Pro-2:hw2 annachen$ python3 q1.py
test 1
3
5
4
6

test 2
3
2
1

test 3
2
2

test 4
1
1
2
5
10
5
2
```

Step 5: Add Documentation

```
class Stack:
    def __init__(self):
        self.nums = [] # stack initialized as empty

    def push(self, val):
        ''' Adds an integer onto the stack '''
        self.nums.append(val)

    def pop(self):
        '''
        Removes most recently added item from stack and returns it.
        Assumes stack is not empty.
        '''
        return self.nums.pop()

    def peek(self):
        '''
        Returns most recently added item from stack without removing it.
        Assumes stack is not empty.
        '''
        return self.nums[-1]
```

Step 6: Extra Test Cases Used

```
st2.push(1)
st2.push(2)
st2.push(3)
st2.pop()   (delete and return 3)
st2.pop()   (delete and return 2)
st2.pop()   (delete and return 1)
```

```
st3.push(1)
st3.push(2)
st3.peek()  (return 2)
st3.peek()  (return 1)
```

```
st4.push(1)
st4.peek()  (return 1)
st4.pop()   (delete and return 1)
st4.push(2)
st4.peek()  (return 2)
st4.push(5)
st4.peek()  (return 5)
st4.push(10)
st4.pop()   (delete and return 10)
st4.pop()   (delete and return 5)
st4.pop()   (delete and return 2)
```

2 C++

Step 2: Actual Code

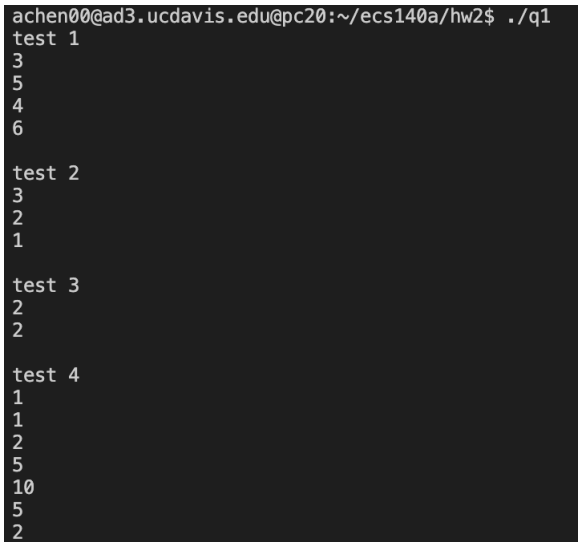
```
class Stack {
    public:
        std::vector<int> nums;

        void push(int val) {
            nums.push_back(val);
        }

        int pop() {
            int last_val = nums.back();
            nums.pop_back();
            return last_val;
        }

        int peek() {
            return nums.back();
        }
};
```

Step 3: Working Code

There were no syntax errors, so the initial working code was the same as the previous step.

Step 4: Debug Process

All test cases behaved as expected.



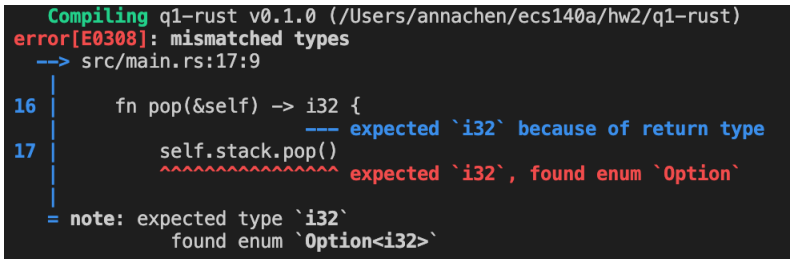
Step 5: Add Documentation

```
class Stack {
    public:
        std::vector<int> nums; // empty vector

        /* Adds an integer onto the stack */
        void push(int val) {
            nums.push_back(val);
        }

        /**
         * Remove most recently added item from stack and return it.
         * Assumes stack is non-empty.
         */
        int pop() {
            int last_val = nums.back();
            nums.pop_back();
            return last_val;
        }

        /**
         * Returns most recently added item from stack without removing it.
         * Assumes stack is non-empty.
         */
        int peek() {
            return nums.back();
        }
};
```

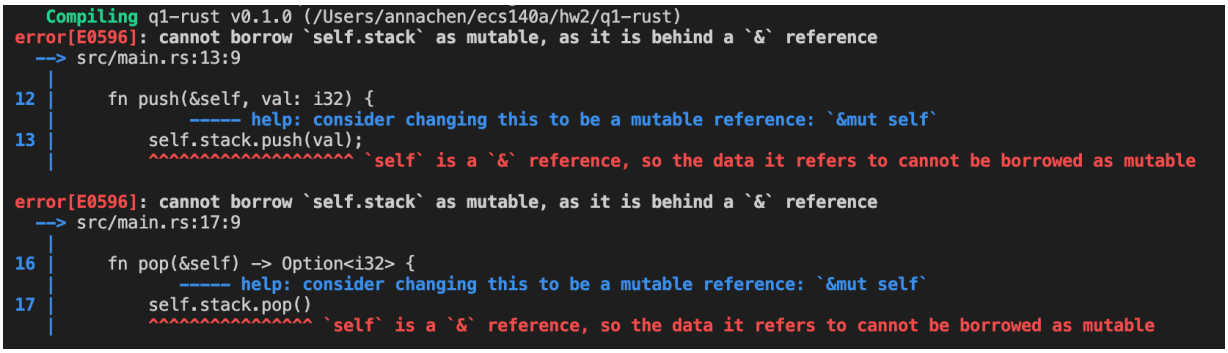
3 Rust

Step 2: Actual Code

```
struct Stack {
    stack: Vec<i32>
}

impl Stack {
    fn new() -> Stack {
        Stack {
            stack: Vec::new()
        }
    }

    fn push(&self, val: i32) {
        self.stack.push(val);
    }

    fn pop(&self) -> i32 {
        self.stack.pop()
    }

    fn peek(&self) -> i32 {
        self.stack[self.stack.len()-1]
    }
}
```



This error was raised because the actual type of the returned value did not match the expected. To fix this, I changed the return type of `pop()` from `i32` to `Option<i32>`.



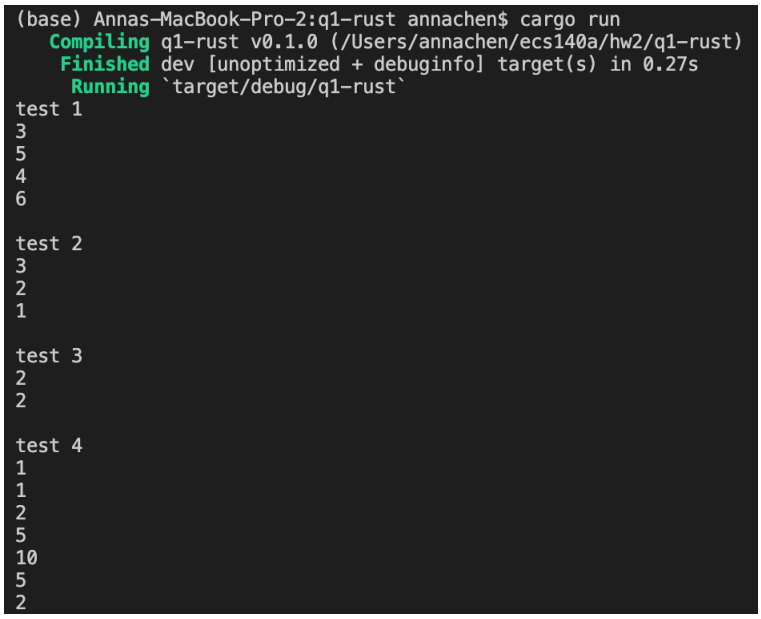
This error was raised because the functions modify the stack but I did not specify self instance to be mutable. To fix, I used the suggested fixes by the compiler.

Step 3: Working Code

```
struct Stack {
    stack: Vec<i32>
}

impl Stack {
    fn new() -> Stack {
        Stack {
            stack: Vec::new()
        }
    }

    fn push(&mut self, val: i32) {
        self.stack.push(val);
    }

    fn pop(&mut self) -> Option<i32> {
        self.stack.pop()
    }

    fn peek(&self) -> i32 {
        self.stack[self.stack.len()-1]
    }
}
```

Step 4: Debug Process

All test cases behaved as expected.



Step 5: Add Documentation

```
struct Stack {
    stack: Vec<i32>
}

impl Stack {
    // initializer
    fn new() -> Stack {
        Stack {
            stack: Vec::new()
        }
    }

    // Adds an integer onto the stack
    fn push(&mut self, val: i32) {
        self.stack.push(val);
    }

    // Removes the most recently added item from stack and returns it
    // Assumes stack is non-empty
    fn pop(&mut self) -> Option<i32> {
        self.stack.pop()
    }

    // Returns most recently added item to stack without removing it
    // Assumes stack is non-empty
    fn peek(&self) -> i32 {
        self.stack[self.stack.len()-1]
    }
}
```