Jake Gendreau
CS 210
Semester Project Phase 2
Stacks

CODE:

```
// Struct defines fields for the type
// We are using a vector as a our base here
struct Stack
{
   stack: Vec<isize>,
}

// Impl defines methods for the type
impl Stack
{
   // Make new stack
   fn new() -> Self
   {
      return Stack {stack: Vec::new()}
   }

   // Pop method
   fn pop(&mut self) -> Option<isize> {
      self.stack.pop() // This automatically returns None if the stack is empty
   }

   // Push method
   fn push(&mut self, item: isize)
   {
      self.stack.push(item);
   }

   // Clone the first value and return
   fn peek(&self) -> Option<&isize>
   {
      return self.stack.last()
   }

   // Return size of the array
   fn size(&self) -> usize
   {
      return self.stack.len();
   }

   // Return bool representing is empty
   fn is_empty(&self) -> bool
   {
      return self.stack.is_empty();
   }

   // Print the stack
```

```
    fn print_stack(&self)
    {
        println!("\nCurrent stack:");
        for item in &self.stack
        {
            println!("{}", item);
        }
    }
}

fn main()
{
    let mut stack: Stack = Stack::new();

    // Show stack.size()
    println!("Stack size using stack.size()");
    println!("{}", stack.size());

    // Show stack.is_empty()
    println!("\nIs stack empty using stack.is_empty()?");
    println!("{}", stack.is_empty());

    // Attempty to pop from empty stack
    println!("\nPopping from empty stack...");
    if let Some(value) = stack.pop()
    {
        println!("Popped {}", value);
    }
    else
    {
        println!("Stack is empty");
    }

    // Attempt to peek empty stack
    println!("\nPeeking empty stack...");
    if let Some(value) = stack.peek()
    {
        println!("Top value is {}", value);
    }
    else
    {
        println!("Stack is empty");
    }

    // Push 1 - 3 to stack
    println!("\nPushing 1, 2, 3 to the stack...");
    for i in 1..=3
    {
        stack.push(i);
    }

    // Print the new stack
```

```
stack.print_stack();

// Show size again with full stack
println!("\nStack size using stack.size()");
println!("{}", stack.size());

// Show is_empty() with full stack
println!("\nIs stack empty using stack.is_empty()?");
println!("{}", stack.is_empty());

// Peek a full stack
println!("\nPeeking stack...");
if let Some(value) = stack.peek()
{
   println!("Top value is {}", value);
}
else
{
   println!("Stack is empty");
}

// Pop a full stack
println!("\nPopping from stack...");
if let Some(value) = stack.pop()
{
   println!("Popped {}", value);
}
else
{
   println!("Stack is empty");
}

// Pop again
println!("\nPopping from stack...");
if let Some(value) = stack.pop()
{
   println!("Popped {}", value);
}
else
{
   println!("Stack is empty");
}

// Pop to empty stack
println!("\nPopping from stack...");
if let Some(value) = stack.pop()
{
   println!("Popped {}", value);
}
else
{
   println!("Stack is empty");
```

```
    }

    // Show that stack is empty using stack.size()
    println!("\nStack size using stack.size()");
    println!("{}", stack.size());

    // And stack.is_empty()
    println!("\nIs stack empty using stack.is_empty()?");
    println!("{}", stack.is_empty());

    // Priint the now empty stack
    stack.print_stack();
}
```

TEST CASES:
1) Create stack
2) Use stack.size() to show the function with an empty stack
3) Use stack.is_empty() to demonstrate with empty stack
4) Attempt to pop() from an empty stack
5) Attempt to peek() from an empty stack
6) Push values 1, 2, 3 and 3 onto the stack in that order
7) Print the stack to show that they are there
8) Use stack.size() with a populated stack
9) Use stack.is_empty() with a populated stack
10) Peek() a populated stack
11) Pop() until the stack is empty again
12) Use stack.size() on an empty stack
12) Use stack.is_empty() on an empty stack
13) Print the now empty stack

This set of testcases covers every scenario which the stack allows - pushing, popping, peeking, sizing, and is_emptying on both a populated and unpopulated stack, as per the requirements in the assignment.

SCREENSHOT OF CODE RUNNING:

```
jake@pop-os:~/Documents/Schoolwork/CS210/SemProject/Phase2$ ./stack
Stack size using stack.size()
0

Is stack empty using stack.is_empty()?
true

Popping from empty stack...
Stack is empty

Peeking empty stack...
Stack is empty

Pushing 1, 2, 3 to the stack...

Current stack:
1
2
3

Stack size using stack.size()
3

Is stack empty using stack.is_empty()?
false

Peeking stack...
Top value is 3

Popping from stack...
Popped 3

Popping from stack...
Popped 2

Popping from stack...
Popped 1

Stack size using stack.size()
0

Is stack empty using stack.is_empty()?
true

Current stack:
jake@pop-os:~/Documents/Schoolwork/CS210/SemProject/Phase2$
```