Vectors

Vectors are like arrays but they are stored in heap, correspondingly allowing us to expand and shrink it in size.

Program 1

```
fn main() {
        let v = vec![1,2,3];
        println!("{}",v); //can’t be printed with a default formatter
}
```

Program 2

```
fn main() {
        let v = vec![1,2,3,'a']; // expected integer found char
}
```

Program 3

```
fn main() {
        let v = vec!['a',1,2,3]; //expected char found integer
}
```

Program 4

```
fn main() {
        let v:Vec<i32>;
        v = vec![1,2,3];
        println!("{:?}",v);
}
```

Reading Elements of Vectors

Program 5

```
fn main() {
        let v = vec![1, 2, 3, 4, 5];
        let third: &i32 = &v[2];
        println!("The third element is {}", third);
        println!("{:?}", v.get(2));
}
```

Program 6

```
fn main() {
	let v = vec![1, 2, 3, 4, 5];
	let third: &i32 = &v[2];
	println!("The third element is {}", third);
	match v.get(2) {
		Some(third) => println!("The third element is {}", third),
		None => println!("There is no third element."),
	}
}
```

Program 7

```
fn main() {
	let v = vec![1, 2, 3, 4, 5];
	let does_not_exist = &v[100]; //panic
	let does_not_exist = v.get(100);
}
```

Program 8

```
fn main() {
	let mut v = vec![1, 2, 3, 4, 5];
	let first = &v[0];
	v.push(6);
	println!("The first element is: {}", first);
}
```

This error is due to the way vectors work: adding a new element onto the end of the vector might require allocating new memory and copying the old elements to the new space, if there isn't enough room to put all the elements next to each other where the vector currently is. In that case, the reference to the first element would be pointing to deallocated memory. The borrowing rules prevent programs from ending up in that situation.

Iterating over the Values in a Vector

Program 9

```
fn main() {
        let v = vec![100, 32, 57];
        for i in v {
        println!("{}", i);
        };
        println!("{:?}",v); //value has been moved
}
```

Program 10

```
fn main() {
        let v = vec![100, 32, 57];
        for i in &v {
        println!("{}", i);
        };
        println!("{:?}",v);
}
```

Program 11

```
fn main() {
        let mut v = vec![100, 32, 57];
        for i in &mut v {
        *i += 50; //dereferencing a pointer
        }
        println!("{:?}",v);
}
```