P1 Serie 7

Abdihakin Sahal Omar 20-947-107
Manuel Flückiger 22-11-502

December 9, 2022

Excercise 1

a True

b False

c False

d True

e True

f False

Excercise 2

```
for i = 1 to 100
    if (i is divisible by 3 AND i is divisible by 5)
        output "BizzBuzz"
    else if (i is divisible by 3)
        output "Bizz"
    else if (i is divisible by 5)
        output "Buzz"
    else output i
    end if
end for
```

Excercise 3

```
Input: Arrays A and B of length n
Output: Array C of length n + 1
Function BinarySum (A, B, n)
    Let carry = 0
    Let C[n+1] // Initialize output array
    For i = 0 to n-1
        C[i] = (A[i] + B[i] + carry) % 2 // Calculate sum of current bit
        carry = (A[i] + B[i] + carry) / 2 // Calculate carry for next iteration
```

```
    End For
    C[n] = carry // Store carry in the n-th bit of C
    Return C
End Function
```

Excercise 4

```java
public static <T> void shuffle(List<T> list) {
    int n = list.size() - 1;
    Random rand = new Random();
    for(int i = n; i > 0; --i) {
        int r = rand.nextInt(i+1);
        List<T> copy = new LinkedList<>(list);
        list.set(i, copy.get(r));
        list.set(r, copy.get(i));
    }
}
public static void swap(List<Integer> list, int i, int r) {
    int temp = list.get(i);
    list.set(i, list.get(r));
    list.set(r, temp);
}
```

Excercise 5

```
Java 1.0
Python 1.1
12345
```

Excercise 6

The stack s will look like this: s = [5, 72, 37]

Excercise 7

The stack s will look like this: s = [72, 37, 15]

Excercise 8

```java
    public boolean set(int index, Object object) {
        if(index > size - 1)
            return false;
        else {
            listElements[index] = object;
            return true;
    }
}
```

Excercise 9

```java
    public int size() {
    if (this.startNode == null)
      return 0;
    else {
      int count = 1;
      Node<E> temp = this.startNode;
      while(temp.getNext() != null) {
        temp = temp.getNext();
        count++;
      }
      return count;
```

```
12      }
13      }
14
```

Excercise 10

Die Java-collection ist ein abstrakter Typ, der die gemeinsamen Verhaltensweisen und Merkmale verschiedener Sammlungen definiert. Die Get- und Set-Methoden sind spezifisch für die Implementierung der Sammlung und werden daher nicht in der Schnittstelle angegeben. Jede Implementierung der Schnittstelle kann die get- und set-Methoden anders implementieren, so dass es dem Benutzer der Sammlung obliegt, die Implementierung der von ihm verwendeten Sammlung zu kennen und zu wissen, wie die get- und set-Methoden richtig zu verwenden sind. Eine Liste hat eine bestimmte Reihenfolge. Ein Set hat keine, also ist set.get(i) kein sinnvolles Konzept. Es wäre sinnvoll, eine get(i)-Operation für SortedSet und TreeSet zu haben, aber es gibt sie nicht.