# Lab 3: Sets and Bits Manipulation

Team members:

Adham Anas – 22010601

Badr ElSayed - 22010664

## Part 1: Basic Bit Operations:

### 1. Problem statement:

Implement 4 bit operations. Index of position starts from 0 at the least significant bit of the number:

1) Get the bit of a number at certain position (bit at position 1 of number 10 (1010) is 1.
2) Set the bit of a number at certain position -make it 1-, and return the number after changes.
3) Clear the bit of a number at certain position -make it 0-, and return the number after changes.
4) Update the bit of a number at certain position -make 1 or 0 based on the passed Boolean value-, and return the number after changes.

### 2. Used data structures:

1) Int: for the number and position of the bit.
2) Boolean: for the passed “value” in the updateBit() function, that determines whether to set the bit or clear the bit.

## 3. Sample runs and different test cases

1) getBit

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 1
Enter the number: 2
Enter the position: 0
Bit at position 0 is 0
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 1
Enter the number: 2
Enter the position: 1
Bit at position 1 is 1
```

2) getBit

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 1
Enter the number: 10
Enter the position: 3
Bit at position 3 is 1
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 1
Enter the number: 10
Enter the position: 2
Bit at position 2 is 0
```

3) clearBit

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 2
Enter the number: 10
Enter the position: 1
Number after clearing bit: 8
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 2
Enter the number: 10
Enter the position: 0
Number after clearing bit: 10
```

4) setBit

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 3
Enter the number: 12
Enter the position: 0
Number after setting bit: 13
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 3
Enter the number: 12
Enter the position: 1
Number after setting bit: 14
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 3
Enter the number: 12
Enter the position: 2
Number after setting bit: 12
```

5) updateBit

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 4
Enter the number: 8
Enter the position: 1
Enter the new value (true for 1, false for 0): true
Number after updating bit: 10
```

```
Choose an operation:
0. Exit
1. Get Bit
2. Clear Bit
3. Set Bit
4. Update Bit
Enter your choice: 4
Enter the number: 8
Enter the position: 3
Enter the new value (true for 1, false for 0): false
Number after updating bit: 0
```

## 4. Assumptions and details:

There's a never-ending user-input loop, unless user exits by entering '0'.

## Part 2: Sets Operations using Bits Manipulation:

### 1. Problem statement:

Set data structure that takes in the constructor a **list of strings as a Universe (U)**. The elements in a Set are subset of U. **bits are used to represent the set**. The Set data structure has the following main operations:

1) Add string to a set
2) Union with another set
3) Intersection with another set
4) Complement of a set
5) Difference from another set
6) Cardinality of a set
7) Get elements of a set

8) Get elements of the Universe
9) Is an element in the Universe

### 2. Used data structures:

1) ArrayList:
   a. Store the elements of the Universe, as strings.
   b. Return a set (in case of printing a set) stored in an ArrayList, as strings.
2) HashMap:
   a. Store each set name with its bitwise value, ex: {A:0100} , {B:1011} (binary for visualization only, they are stored 'int')
   b. Used as a helper to remove repeated elements in a list (Universe)
3) Int:
   Indices, store the sets values, etc..
4) String:
   Set names, set/universe elements, etc..

5) BitsManipulator (Part 1 Class):
   a. For setting bits, in the function addToSet: adds an element to a certain set by setting the bit that corresponds to the index of the element in the Universe to 1.

## 3. Sample runs and different test cases:

First: program sequence before choosing 1 of the 6 operations:

```
Enter the universe set as a list of words separated by commas ( a, b, c ):
red, , , blue, red, 1, 2, 3, ball, car, 4, blue
```

```
Universe: [red, blue, 1, 2, 3, ball, car, 4]
Enter the number of sets:
3
```

```
Universe: [red, blue, 1, 2, 3, ball, car, 4]
Enter elements of set 1 separated by commas:
red, 1, 2, car
```

```
[red, 1, 2, car]
Universe: [red, blue, 1, 2, 3, ball, car, 4]
Enter elements of set 2 separated by commas:
red, blue, 1, 2, 4
```

```
[red, blue, 1, 2, 4]
Universe: [red, blue, 1, 2, 3, ball, car, 4]
Enter elements of set 3 separated by commas:
red, blue, ball, car
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt:
```

Last Result: (Output of last operation)

Now: each operation:

1) Union
   i.

```
Choose the 2 sets separated by a comma (ex: 1, 3):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
1, 2
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Union Answer: [red, blue, 1, 2, car, 4]
```

ii.

```
Choose the 2 sets separated by a comma (ex: 1, 3):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
2, 3
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Union Answer: [red, blue, 1, 2, ball, car, 4]
```

2) Intersection

i.

```
Choose the 2 sets separated by a comma (ex: 1, 3):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
1, 3
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Intersection Answer: [red, car]
```

ii.

```
Choose the 2 sets separated by a comma (ex: 1, 3):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
2, 2
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Intersection Answer: [red, blue, 1, 2, 4]
```

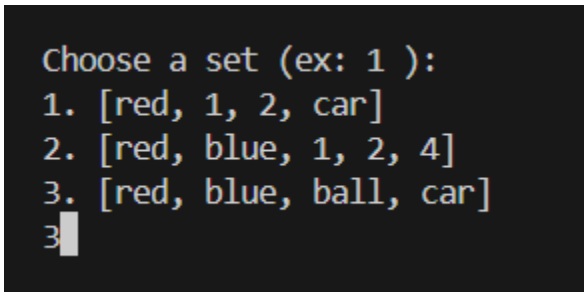
3) Complement

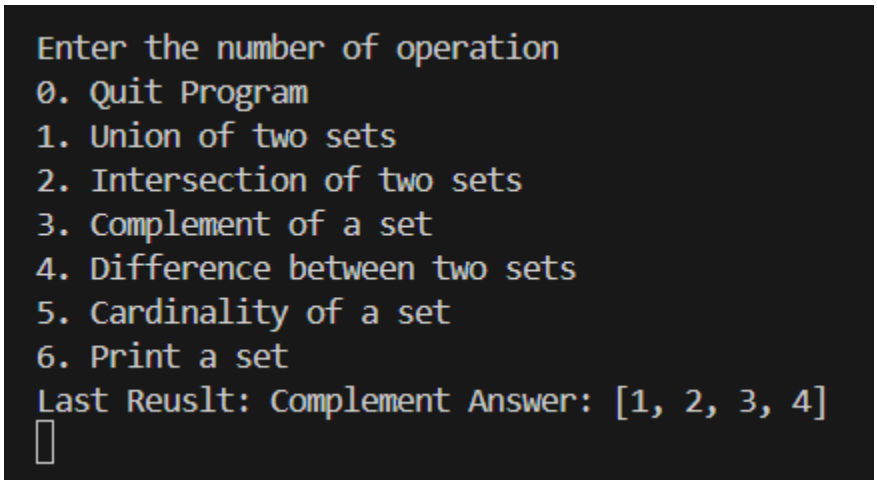
i.

```
Choose a set (ex: 1 ):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
1
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Complement Answer: [blue, 3, ball, 4]
```

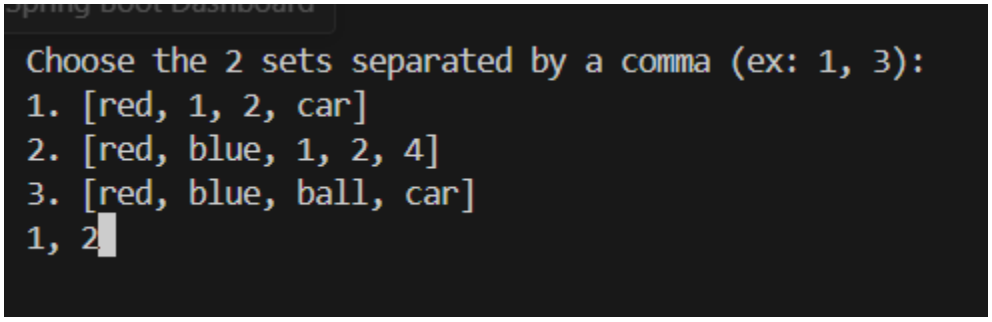
ii.




4) Difference

i.



```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Difference Answer: [car]
```

ii.

```
Choose the 2 sets separated by a comma (ex: 1, 3):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
3, 3
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Difference Answer: []
```

5) Cardinality

```
Choose a set (ex: 1 ):
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
2
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Cardinality Answer: 5
```

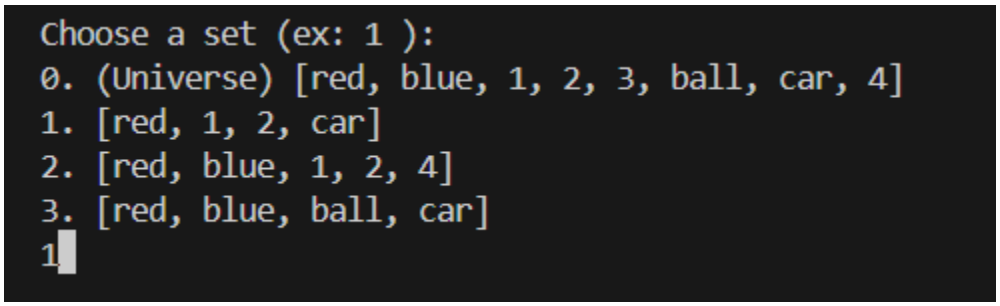
6) Print

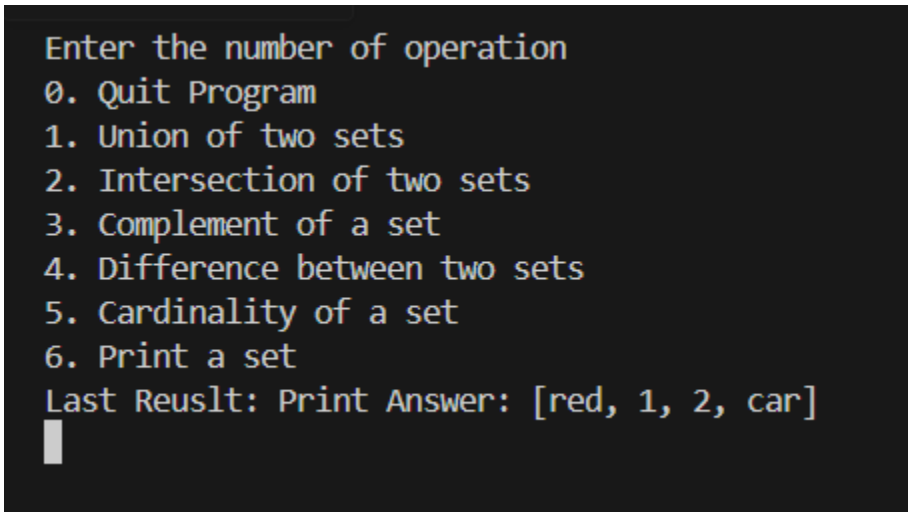
i.

```
Choose a set (ex: 1 ):
0. (Universe) [red, blue, 1, 2, 3, ball, car, 4]
1. [red, 1, 2, car]
2. [red, blue, 1, 2, 4]
3. [red, blue, ball, car]
0
```

```
Enter the number of operation
0. Quit Program
1. Union of two sets
2. Intersection of two sets
3. Complement of a set
4. Difference between two sets
5. Cardinality of a set
6. Print a set
Last Reuslt: Print Answer: [red, blue, 1, 2, 3, ball, car, 4]
```

ii.




## 4. Assumptions and details:

There's a never-ending user-input loop, unless user exits by entering '0'.
The program loop is as follow:
(Each stage only proceeds if correct input is entered)

1) Program asks for user input for the universe
2) Then, asks for the number of subsets (Must be at least 1)
3) Asks for elements of each set
4) Here is the operations loop, it never ends unless user enters 0.

Run Part1.java for Part 1 Problem
Run Part2.java for Part 2 Problem