# Module 01

## Public boolean LoadPuzzle(BufferedReader stream)

- This method checks if the input file which is a BufferefReader, can be successfully read, has no error like blank line and if has blank line then returns false.

**Test 01**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
b 1 =
c 50 *
d 1 -
e 10 +
f 20 *
g 15 +
h 24 *
A 8 +
```

**Actual Result: true**

- When this input is read passed through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.

**Test 02**

**Input:**

```
5
abcdd

accef
gggef
gheef
hhAAA
a 2 /
b 1 =

c 50 *
d 1 -
e 10 +
f 20 *
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **false** as this input is a not **valid** one as there are multiple blank lines throughout the file i.e. between 2nd & 3rd and 8th & 9th lines.

**Test 03 (boundary cases)**

**Input:**

**“ ” //( empty file)**

**Actual Result: false**

- When this input is read passed through LoadPuzzle(), it will return **false** as this input file is empty.

# Module 02

## Public boolean LoadPuzzle(BufferedReader stream)

## Public boolean validate()

- Validate() method verify if the input file successfully read by LoadPuzzle() satisfies every input validation condition for each section.
- **I**f it validates every condition for each section, then it returns true or else false for any type of error. Validation condition are as follow**-**
    1. String in the first line should be of integer type which gives size of puzzle
    2. String next few lines should be should be of same size as puzzle size and
    3. Lines number after puzzle size + 1 should be of format – "operation_group output_value operation", where operation_group should be single character string, output_value should be an integer and operation must be one from list – (*,+,=,/,-). Note that any **empty/null** value in this variable will be processed as **invalid** input.

**Test 01**
**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
b 1 =
c 50 *
```

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: true**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **true** as the first string is of integer and assigned as puzzle size, next 5 lines are string of length puzzle size and has no gap ( every cell of similar group is connected adjacently) and remaining lines satisfies "operation_group should be single character string, output_value should be an integer and operation must be one from list – (*,+,=,/,-)" condition.

**Test 02**
**Order of calling methods**

3. **loadPuzzle()**
4. **Validate()**

**Input:**

5.0

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** as the first string is of float data type.

**Test 03**
**Order of calling methods**

5. **loadPuzzle()**
6. **Validate()**

**Input:**

-5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** as the first string is a negative integer.

**Test 04**
**Order of calling methods**

7. **loadPuzzle()**
8. **Validate()**

**Input:**

5

abadd

accef

ggge

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** as one of the 5 lines of string is not of length same as puzzle size and has gap in group "a" ( every cell of similar group is connected adjacently).

**Test 05**

**Order of calling methods**

9. **loadPuzzle()**
10. **Validate()**

**Input:**

5

abcdd

accef

gggef

gh eef

hhA AA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: true**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **true** as the first string is of integer and assigned as puzzle size, next 5 lines are string of length puzzle size and has no gap ( every cell of similar group is connected adjacently) because even though 2 strings as whitespace in between, I have allowed white spaces for this section and ignoring it by removing it during validation.

**Test 06**
**Order of calling methods**

11. **loadPuzzle()**
12. **Validate()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
  1 =
c 50 *
d   -
e 10 +
f 20
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** because in section 3 of file, there is a null group value in 8th line, **empty** value in output in 10th line and empty operation value in 12th line.

**Test 07**

**Order of calling methods**

13. **loadPuzzle()**
14. **Validate()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
 b  1 =
c 50 *
d  1.67  -
e 10 +
f 20  *
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.

- This input is then passed to validate() and it returns **false** because in section 3 of file, value in output in 10th line is of float type.

**Test 08**
**Order of calling methods**

15. **loadPuzzle()**
16. **Validate()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
 b  1 =
c 50 *
d  1  -
e -10 +
f 20  *
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** because in section 3 of file, value in output in 11th line is a negative integer.

**Test 09 (Boundary Case)**
**Order of calling methods**

17. **loadPuzzle()**
18. **Validate()**

**Input:**

```
-5
abadd
accef
ggge
gheef
hhAAA
a 2 /
  1 =
c 50 *
d   -
e 10 +
f 20
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- When this input is read through LoadPuzzle(), it will return **true** as this input is a **valid** one without blank lines in between each string line.
- This input is then passed to validate() and it returns **false** because first string is of float data type, in next 5 lines some strings are not of length same as puzzle size and there is gap in group "a" and in section 3 of file, there is a null group value in 8th line, **empty** value in output in 10th line and empty operation value in 12th line.

**Test 10**

**Order of calling methods**

19. **Validate()**
20. **loadPuzzle()**

**Input:**

```
-5
abadd
accef
ggge
gheef
hhAAA
a 2 /
  1 =
c 50 *
d   -
e 10 +
f 20
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- it will return **false** as the validate() is executed before loadPuzzle()

# Module 03

## Public boolean LoadPuzzle(BufferedReader stream)

## Public boolean validate()

## Public boolean solve()

- Solve() method checks if the input file which is read by loadPuzzle() successfully and validated by validate(), can be solved or not.
- Solve() will always return false if either loadPuzzle() or validate() return false.

**Test 01**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
b 1 =
c 50 *
d 1 -
e 10 +
f 20 *
g 15 +
h 24 *
A 8 +
```

**Actual Result: true**

- For this **valid** input, loadPuzzle() and validate() will return **true** and solve() will also return **true** as this puzzle is solvable.

**Test 02**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**

**Input:**

```
5
abcdd
accef
gggef
gheef
hhAAA
a 2 /
b 1
c 50 *
d 1 -
e 10 +
f 20 *
g 15 +
h 24 *
A 8 +
```

**Actual Result: true**

- For this in**valid** input, loadPuzzle() will return true but validate() will return **false** and solve() will also return **false** as validate() is not true.

**Test 03 (Boundary case)**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**

**Input:**

```
-5


abadd
accef
ggge
gheef
hhAAA
a 2 /
  1 =
c 50 *
d   -
e 10 +
f 20
g 15 +
h 24 *
A 8 +
```

**Actual Result: true**

- For this **invalid** input, loadPuzzle() and validate() will return **false** and solve() will also return **false.**

**Test 04**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**

**Input:**

```
5
abcdd
accef
ggeef
gheef
hhAAA
a 2 /
b 1
c 50 *
d 1 -
e 10 +
f 20 *
g 15 +
h 24 *
A 8 +
```

**Actual Result: false**

- For this **valid** input, solve() will always return **false** as there is not possible solution available.

**Test 05**

**Order of calling methods**

1. **Solve()**
2. **loadPuzzle()**
3. **Validate()**

**Input:**

5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: true**

- For this **valid** input, solve() will always return **false** as it’s called before validate() and loadPuzzle().

## Module 04

**Public boolean LoadPuzzle(BufferedReader** stream**)**

**Public boolean validate()**

**Public boolean solve()**

**Public string print()**

- Print() method will return a string of solution matrix if solve() return true which means puzzle was successfully solved and false if no solution possible.

**Test 01**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**
4. **Print()**

**Input:**

5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: “14352\n32514\n53421\n25143\n41235\n”**

- For this **valid** input, loadPuzzle(), validate() and solve() will return **true** as this is a valid puzzle and when print is called, a string is returned which stores value of each row separated by “\n”.

**Test 02**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Solve()**
4. **Print()**

**Input:**

5

abcdd

accef

gggef

ggeef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: “14352\n32514\n53421\n25143\n41235\n”**

- For this **valid** input, loadPuzzle(), validate() will return **true** but solve() will return **false** as this is a valid puzzle but no solution is possible and when print is called, a **empty** string is returned.

**Test 02**

**Order of calling methods**

1. **Print()**
2. **loadPuzzle()**
3. **Validate()**
4. **Solve()**

**Input:**

5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: “”**

- For this **valid** input, print() returns **empty** string as the solution matrix is empty because print is called before solve() can figure out a solution.
- Print() will always return **empty** string if executed before loadpuzzle(), validate() or solve().

# Module 05

## Public boolean LoadPuzzle(BufferedReader stream)

## Public boolean validate()

## Public boolean solve()

## Public int choices()

- choices() method will return a integer value which indicates the count of traversals happened during executing solve() before a solution was reached.

**Test 01**

**Order of calling methods**

5. **loadPuzzle()**
6. **Validate()**
7. **Solve()**
8. **Choices()**

**Input:**

5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: “233”**

- For this **valid** input, choices() will return integer value **233** which shows total 233 traversal occurred during solve() before solution was reached.

**Test 02**

**Order of calling methods**

1. **loadPuzzle()**
2. **Validate()**
3. **Choices()**
4. **Solve()**

**Input:**

5

abcdd

accef

gggef

gheef

hhAAA

a 2 /

b 1 =

c 50 *

d 1 -

e 10 +

f 20 *

g 15 +

h 24 *

A 8 +

**Actual Result: “0”**

- For this **valid** input, choices() will return **0** ,as it is called before executing solve() and will always return **0** if called even before loadPuzzle() or validate().