WIA1002 Data Structure
Lab 5: Stack

1. Ahmad randomly selects a number of candy and insert into a candy container in the figure below. The candies are **blue, orange, red** and **yellow** in colour. His brother Ali likes to eat the blue ones. Ali takes all the candies one by one from the container and eats the blue ones. Then, he put back the rest of candies in order. Write a program to simulate the process using stack with linked list.

Example output:

```
The candies in the container:
 <-- Blue <-- Yellow <-- Yellow <-- Orange <-- Blue <-- Red <-- Orange
Ali takes all the candies one by one from the container and eats the blue ones.
He puts back the rest of candies in the container.
The candies in the container:
 <-- Yellow <-- Yellow <-- Orange <-- Red <-- Orange
```

2. Write a postfix calculator that can compute the integer value of an infix expression. The operators used in the calculator are **add, sub, mul, div, mod**. Extra operators include **ob**=open bracket and **cd**=close bracket.

Infix to Postfix Conversion
When an operand is read, output it
When an operator is read
  pop until the stack has a lower precedence operator
  then, push the new operator
When (is found, push.
When) is found, pop until the matching (.
When reach the end of input, pop until the stack is empty.

Postfix Evaluation
Scan the postfix expression from left to right
If recognize an operand, push it on the stack.
If recognize an operator, pop two operands from the stack.
Apply the operation and push the result on the stack.
Continue the expression until the end. The result is on the top of the stack.

Example output:

```
Enter infix expression : 34 add 11 mul 7
The infix expression is: 34 + 11 * 7
The postfix expression is: 34 11 7 * +
The result is: 111

Enter infix expression : ob 300 mul 75 add 4350 cb div 15 add 1126
The infix expression is: ( 300 * 75 + 4350 ) / 15 + 1126
The postfix expression is: 300 75 * 4350 + 15 / 1126 +
The result is: 2916

Enter infix expression : ob ob 5 add 8 cb sub ob 7 sub 3 cb cb mul 8 add 25 mod 3
The infix expression is: ( ( 5 + 8 ) - ( 7 - 3 ) ) * 8 + 25 % 3
The postfix expression is: 5 8 + 7 3 - - 8 * 25 3 % +
The result is: 73
```

3. Write a program to solve the N-Queen problem.

```
Create empty stack
Set current position to 0
while true
   Iterate from the current position to N
      if there is a valid position
         push the position to stack     // get the column
         set the current position to 0
         set the position as valid

   if there is no valid position
      if stack is empty
         exit the while loop // stop search
      otherwise
         current position = pop stack
         set current position to next position //backtracking

   if stack has size N  // found a solution
      display the solution by popping the stack
      set current position to the last element of the stack
      set current position to next position // look for next solution
```

Example output:

```
Solving the 4 Queens problem
* * Q *
Q * * *
* * * Q
* Q * *

* Q * *
* * * Q
Q * * *
* * Q *

The number of solutions are : 2
```

4. Write a program to evaluate the statement source code correctly matches left-right parentheses, brackets and braces (), [] and {}. The statement will ignore the parentheses, brackets and braces if the \ character appear before parentheses, brackets and braces, \(, \] and etc.

Example output:

```
Enter a expression : System.out.print(str.charAt(3);
System.out.print(str.charAt(3);
                               ^ Missing )


Enter a expression : System.out.print("\( No Error");
System.out.print("\( No Error");
The expression is balanced


Enter a expression : int[] num = 1, 2};
int[] num = 1, 2};
                ^ Extra }

Enter a expression : if (arr[{3] < 4)
if (arr[{3] < 4)
          ^ Missing }
```

5. An XML document contains XML Elements. An XML element begins with <elementName> and ends with </elementName>. An element can contain text and other elements. The element name is case sensitive. (Note: an XML document can only have one root element) Write a program to validate whether an XML document is valid; assume that the element name is valid. Create the xml document using notepad and save as **test.xml.**

   Example output:

```
Contents of XML file
<note>
<test>Sample</test>
<note>Data</note>
</note>
Duplicate root element <note>

Contents of XML file
<note>
<ID>1002</ID>
<Name>Data Structure</name>
</note>
Begin Element : <Name> Invalid ending element : </name>

Contents of XML file
<note>
<ID>1002</ID>
<Name>Data Structure</Name>
</Lecturer>Ang<Lecturer>
</note>
Begin Element : <note> Invalid ending element : </Lecturer>

Contents of XML file
<note>
<ID>1002</ID>
<Name>Data Structure</Name>
<Day>Tuesday</Day>
</note>
The xml document is valid.
```

6. A maze is simply a matrix of cells that are adjacent to one another. The user begins in a special start cell and seeks a path of adjacent cells that lead to the finish cell. A Position is used to identify a unique location within a maze. Any Position can be transformed into another Position by moving to the north, south, east, or west. The Maze class reads a maze description from a file and generates the appropriate adjacency of cells. # is a wall, S marks the start, F marks the finish. Write a program to find the solution from start to finish.

```
Keep track of the path with a stack of positions

At the current position, extend the path one step by trying to go
east, south, west or south
If none of the neighboring squares is available, backtrack by popping a
position off the stack. (If the stack is empty, declare failure.)

When extending the path, push the new position on the stack.
Test to see if the new position can be reached.
Store indicator in the new position to make it impassable in the future.

If Finish has been reached, output the stack contents as the solution path.
```

Example output:

```
The original maze is
####################
#S#       #F   #   #
# ####### #### # # #
#         #  # ### #
##### ### #        #
#   # #   ####### ##
#   # # ### #   #  #
#   # # #   # # ## #
#     #   #   #    #
####################

The Solution is
####################
#S#       #F...#   #
#.####### ####.# # #
#.........#  #.### #
##### ###.#   .... #
#   # #...#######.##
#   # #.### #...#..#
#   # #.#...#.#.##.#
#     #...#...#....#
####################
```