Instruction to candidates:

Your program code and output for each of Task 1 to N should be saved in a single `.ipynb` file using Jupyter Notebook. For example, your program code and output for Task 1 should be saved as:

```
Task1_<your name>_<centre number>_<index number>.ipynb
```

Make sure that each of your `.ipynb` files shows the required output in Jupyter Notebook.

1 Name your Jupyter Notebook as

```
Task4_<your name>_<centre number>_<index number>.ipynb
```

The task is to implement a stack data structure and use it to implement a function to find the position of the nearest smaller value of each integer in an integer list.

For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In [1] :    # Task 4.1
            Program code

            Output:
```

Task 4.1

Implement the class `myStack` which has the following attributes and methods.

Identifier	Data Type	Description
`maxSize`	`INTEGER`	An integer used to store the maximum number of items on the stack.
`stk`	`ARRAY[1:maxSize] OF INTEGER`	An integer array of size equal to `maxSize`. This is used to store the actual integer values on the stack.
`top`	`INTEGER`	Index position in `stk` for the integer stored on the top of the stack. A value of `-1` means that there is no item stored on the stack.

Functions	Descriptions
`push(value)`	This operation pushes `value` onto the top of the stack.
`pop()`	This operation removes and returns the integer value stored on the top of the stack.
`peek()`	This operation returns the integer value stored on the top of the stack.
`isEmpty()`	This operation returns `True` if the stack is empty and `False` otherwise.

[8]

Task 4.2

Test your implementation by writing program code to run three test cases. For each test case, state clearly the test objective using comments.

[3]

Task 4.3

The pseudo-code for the function which returns the position of the nearest previous smaller value of each integer in an integer list as a list is provided in "Task4_3.txt".

Here is an explanation of what the pseudocode does based on the following sample run:

```
lst = [2, 6, 4, 7, 8, 3, 1, 9]
findPosOfNearestPreviousSmallerValue(lst, True)
```

The Boolean argument `True` causes the integer list to be scanned from the left to the right.

Considering the integers in `lst` one by one from left to right:

- 1st value from left is `2`. No previous value on left, the position returned is `-1`.
- 2nd value from left is `6`. Nearest previous smaller value on left is `2`, at position `0`.
- 3rd value from left is `4`. Nearest previous smaller value on left is also `2`, at position `0`.
- 4th value from left is `7`. Nearest previous smaller value on left is `4`, at position `2`.
- 5th value from left is `8`. Nearest previous smaller value on left is `7`, at position `3`.
- 6th value from left is `3`. Nearest previous smaller value on left is `2`, at position `0`.
- 7th value from left is `1`. No value smaller on left, the position returned is `-1`.
- 8th value from left is `9`. Nearest previous smaller value on left is `1`, at position `6`.

Hence the return array will be `[-1, 0, 0, 2, 3, 0, -1, 6]`.

In a second sample run:

```
lst = [2, 6, 4, 7, 8, 3, 1, 9]
findPosOfNearestPreviousSmallerValue(lst, False)
```

The Boolean argument `False` causes the integer list to be scanned from the right to the left.

Considering the integers in `lst` one by one from right to left:

- 1st value from right is `9`. No previous value on right, the position returned is `-1`.
- 2nd value from right is `1`. No previous smaller value on right, the position returned is `-1`.
- 3rd value from right is `3`. Nearest previous smaller value on right is `1`, at position `6`.
- 4th value from right is `8`. Nearest previous smaller value on right is `3`, at position `5`.
- 5th value from right is `7`. Nearest previous smaller value on right is `3`, at position `5`.
- 6th value from right is `4`. Nearest previous smaller value on right is `3`, at position `5`.
- 7th value from right is `6`. Nearest previous smaller value on right is `4`, at position `2`.
- 8th value from right is `2`. Nearest previous smaller value on right is `1`, at position `6`.

Hence the return array will be `[6, 2, 5, 5, 5, 6, -1, -1]`.

Write program code to implement the function `findPosOfNearestPreviousSmallerValue`.

[3]

Test your implementation with the following commands:

```
print("For task 4.3")
lst = [2,4,8,10,12,16,18,9,7,3,1]
print("lst",lst)
print("LtoR",findPosOfNearestPreviousSmallerValue(lst, True))
print("RtoL",findPosOfNearestPreviousSmallerValue(lst, False))
```

[1]

Task 4.4

The function `findPosOfNearestSmallerValue(lst)`:

- takes in a list of integers `lst`
- returns the position of the nearest smaller value of each integer in `lst` as a list.

Here is an explanation of what the function `findPosOfNearestSmallerValue` does base on the sample run:

```
lst = [2, 6, 4, 7, 8, 3, 1, 9]
findPosOfNearestSmallerValue(lst)
```

Considering the integers in `lst` one by one (direction is not important here):

- 1st value is `2`. Nearest smaller value is `1`, at position `6`.
- 2nd value is `6`. The two nearest smaller values are `2` and `4`. Take the larger value and its position is `2`.
- 3rd value is `4`. The two nearest smaller values are `2` and `3`. Take the larger value and its position is `5`.
- 4th value is `7`. The two nearest smaller values are `4` and `3`. Take the larger value and its position is `2`.
- 5th value is `8`. The two nearest smaller values are `7` and `3`. Take the larger value and its position is `3`.
- 6th value is `3`. The two nearest smaller values are `2` and `1`. Take the larger value and its position is `0`.
- 7th value is `1`. There is no value smaller than 1, the position returned is -1.
- 8th value is `9`. The nearest smaller value is `1`, at position `6`.

Using the function `findPosOfNearestPreviousSmallerValue` in **Task 4.3**, implement the function `findPosOfNearestSmallerValue(lst)`.

[4]

Test your implementation with the following commands:

```
print("For task 3.4")
lst = [2,4,8,10,12,16,18,9,7,3,1]
print("lst",lst)
print(findPosOfNearestSmallerValue(lst))
print()
```

[1]

Save your Jupyter Notebook for Task 4.