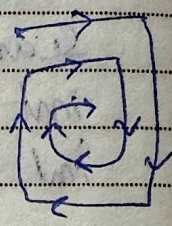
Q4) Take a matrix as input from the user. Search for a given number x and print the indices at which it occurs.

Q5) print Spiral order matrix as output for a given matrix of numbers.

E.g →

Input →

	0	1	2	3	4	5
0	1	5	7	9	10	11
1	6	10	12	13	20	21
2	9	25	29	30	32	41
3	15	55	59	63	68	70
4	40	70	79	81	95	105



output → 1, 5, 7, 9, 10, 11, 21, 41, 70, 105, 95, 81, 79, 70, 40, 15, 9, 6, 10, 12, 13, 20, 32, 68, 63, 59, 55, 25, 29, 30, 29.

Strings

✓ It is a non-primitive data type. It also means that users can define and manipulate the length of it.

✓ There is a class in Java called String. So, we can't use that while defining the ~~variable / Strings for user~~ class. Also, we need to use 'S' instead of 's'.

// String declaration

```
String name = "Tony";
String fullName = "Tony Stark";
```

'new' keyword is used to defined the object 'sc'.

Scanner sc = new Scanner (System.in);

↳ non-primitive data type

String name1 = sc.next();
String name2 = sc.nextLine();

o/p → "Tony Stark" , o/p 1 → Tony,
o/p 2 → Tony Stark.

* When we use sc.next(), it only takes 1 token, which is 'Tony' here, but when we use sc.nextLine(), it takes all the sentences/words used as an i/p.

// Concatenation → attaching 2 Strings.

String firstName = "tony";
String lastName = "stark";

String fullName = firstName + lastName;
s.o.pln (fullName); → tonystark

String fName = firstName + " " + lastName;
s.o.pln (fName); → tony Stark.

* However, the " " (empty String) shall be deleted from memory after the line, because it is not stored explicitly using a variable.

// length of a String

int len = fullName.length(); → gives the total no. of characters in the String.

// charAt method. → Prints each & every character of String.

```
for (int i=0; i<=fullName.length(); i++)
    {
        System.out.println(fullName.charAt(i));
    }
```

// comparison of Strings

```
String name1 = "Tony"
String name2 = "Tony"
if (name1.compareTo(name2)==0)
    {
        S.o.pln("equal");
    }
else
        S.o.pln("not equal");
```

✓ 'compareTo' is the function of comparing one string to another. Here, the ASCII value of each character of the strings are checked.

If S1 > S2, then it returns +ve value.
If S1 = S2, then it returns zero value.
If S1 < S2, then it returns -ve value.

✓ Strings are like objects and so they behave differently in the stored memory location and so, it is better not to use "==" sign in case of strings.

// Substring of a String

Substring (beginning index, end index)

String Sentence = "My name is Tony";

String name = Sentence.substring(11, sentence.length());

✓ The 'substring' function takes 2 parameters - the start index, the end index. However, the end index is not inclusive. So, if it is (0, 4), it will return the substring from index pos. 0 to index pos. 3.

String name1 = sentence.substring(5);

✓ This is also a correct format. It then automatically starts from the mentioned index and goes till the end.

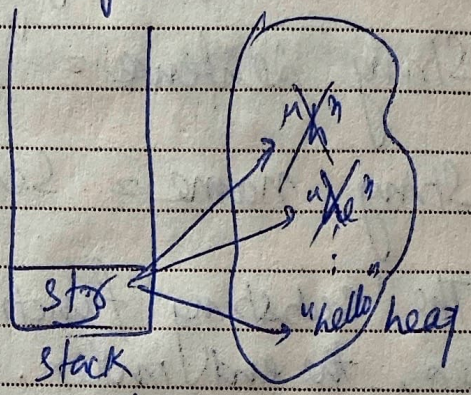
// String immutability.

This is a very important feature. Whenever we define a String, we cannot modify it, delete it or add to it. So, Strings are said to be immutable in Java.

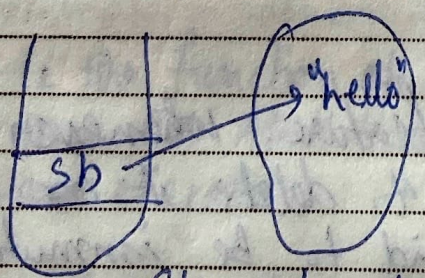
String Builder

- difference b/w String and String Builder

String Str = "h";
Str + "e" → "he";
Str + "l" → "hel";
:
Str + "o" → "hello";



✓ The variable 'str' lies in the Stack memory and the value lies in the heap. once, we change the value, the previous value gets deleted from the heap, and the variable 'str' directly points to the latest value. Now, for a large codebase, where thousands of changes are done on a string, the operation shall cause a delay, which is not at all optimum for user experience.



✓ In case of String builder, it points to a single value in the heap memory. And all the changes are on that value only. So, operational time w.r.t String builder is better compared to Strings.

✓ Also, String builder has it's exclusive operations, which is an added advantage over Strings.

01 April
14th Wk • 091-274
Friday

2022
APRIL
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30
M T W T F S S M T W T F S S

// Declare String builder

```
StringBuilder sb = new StringBuilder("Tony");
// char at index 0.                    → index number
System.out.println(sb.charAt(0));
// Set character at a specific index
sb.setCharAt(0, 'p');
System.out.println(sb);        o/p → Pony
// Inserting char at an index, keeping rest same.
sb.insert(0, 's');
System.out.println(sb);        o/p → Spony
// Deleting any character.
sb.delete(2, 3);               o/p → Spny
          start index, end index
```

Here, the end index is non-inclusive. So, it runs from (start to end-1) index.

// Appending the characters to a String

```
sb.append("a");
sb.append("b");        o/p → Spnyab
```

If compared to normal String, it's not creating a new String and giving it value. It's adding new values to that string only. So, operation is faster.

April
14th Wk • 092-273
02
Saturday

// Length of StringBuilder.

```
Sb.length();
```

O/P → 6.

// Reverse of a String

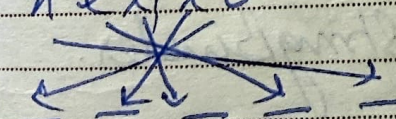
```
StringBuilder sb = new StringBuilder("hello");
for(int i=0; i<sb.length()/2; i++)
{
  int front = i;
  int back = sb.length()-1-i;

  char frontChar = sb.charAt(front);
  char backChar = sb.charAt(back);

  sb.setCharAt(front, backChar);
  sb.setCharAt(back, frontChar);
}
System.out.println(sb);
```

Working

"hello"



✓ Code runs from start to only half of the String.
✓ char at start gets replaced by char at the back & vice versa.

✓ Time Complexity for this algorithm: $O(n/2)$.

✓ There are others methods to perform the same operation. But, this algorithm has the best time complexity

Practice Questions (Strings)

Q) 1) Take an array of Strings input from the user & find the cumulative length of all String.

Q) 2) Input a String from the user. Create a new String called 'result' in which you will replace the letter 'e' in the original String with the letter 'i'.

Example:-
original = "eabcdef" ; result = "iabcdif"
original = "xyz" ; result = "xyz".

Q) 3) Input an email from the user. You have to create an username from the email by deleting the part that comes after '@'. Display the username to the user.

Example:-
email = "apnacollege@gmail.com"
username = "apnacollege".

Q) 4) Repeat Q) 2) using StringBuilder.

Q) 5) Repeat Q) 3) using StringBuilder.

2022

MAY

April

15th Wk • 094-271

04

Monday

box[] = { "apple", "mango", "banana" }

0 1 2

length of box[] = 3

word = apple.

* → when user enters a number and press 'Enter', 1) nextInt() consumes only the number, and not the newline character generated by 'Enter'. Now, if we use nextLine() just after nextInt(), it starts reading from the buffer generated earlier, and not the first word entered by the user.

So, for e.g.,

```
int a = sc.nextInt();
for(int i=0; i<a; i++)
{
  box[i] = sc.nextLine();
}
```

here if a=2 console reads only '1' word and not 2, because it takes into consideration the buffer generated earlier after 2.

To solve :-

```
int a = Integer.parseInt(sc.nextLine());
for(int i=0; i<a; i++)
{
  box[i] = sc.nextLine();
}
```

this would take the i/p as string and then convert it into a number.

So, we don't need to deal with the newline character.