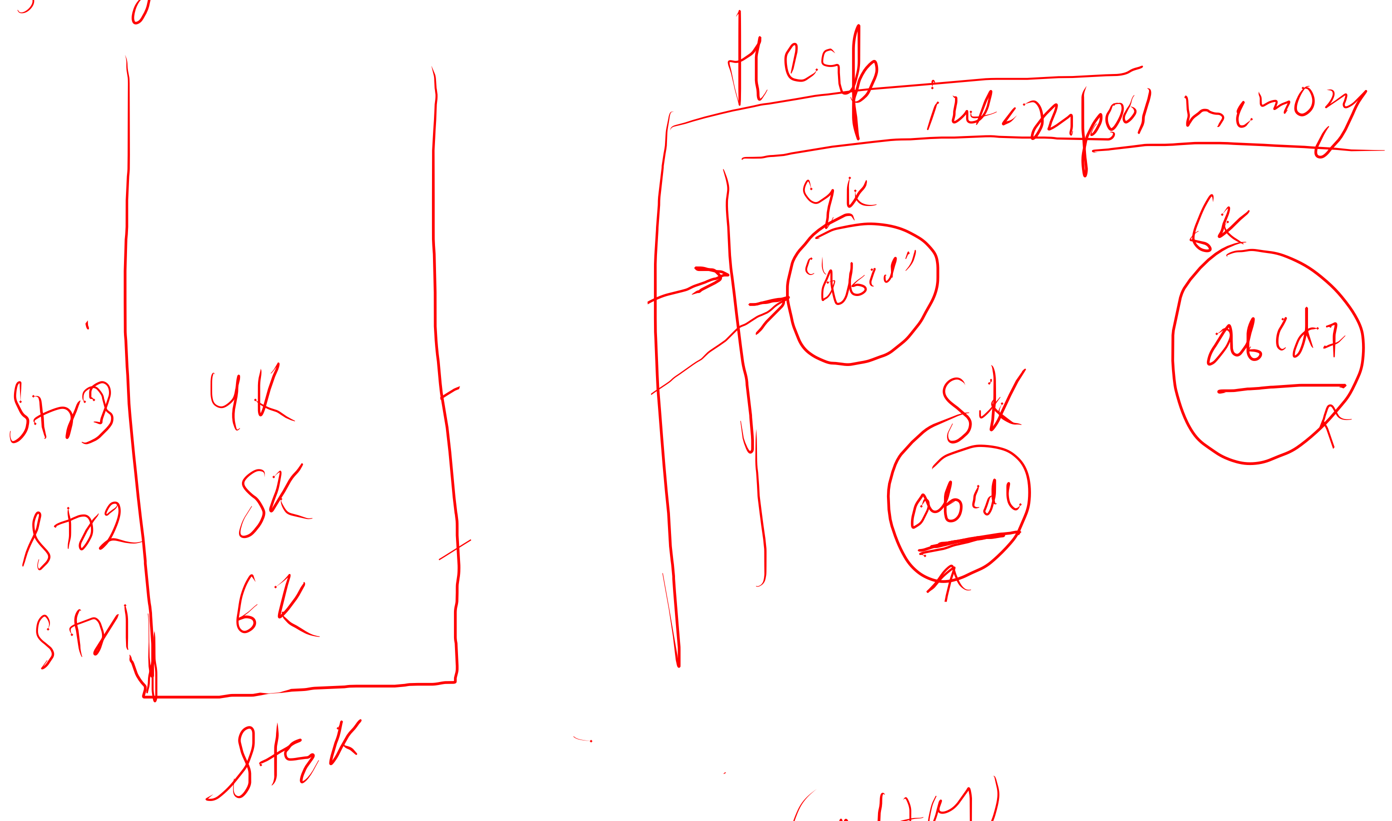
String

N (N+M) M

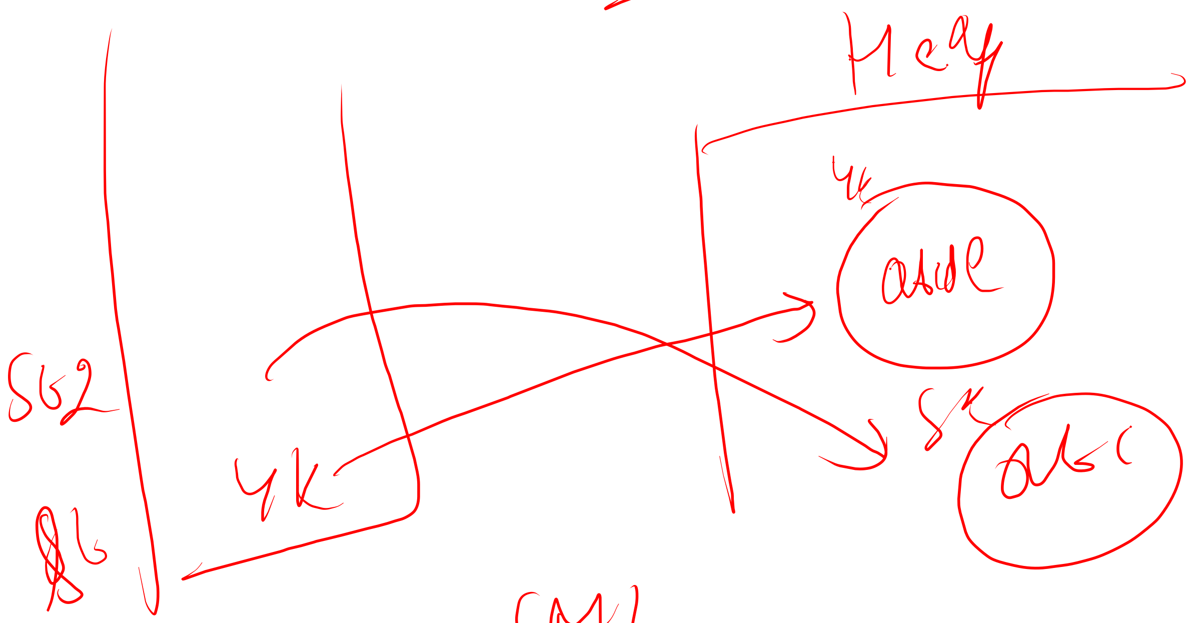
```
String str1= "abcd";
String str2= "abcd";

str2 += 'e';
str1 += '7';
```



(N+M)

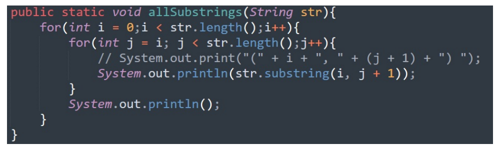
```
Stringbuilder sb = new Stringbuilder();
sb.append("abcd");
sb.append("e");   ← O(1)
```

O(M)



(M)

```
Stringbuilder sb2 = new Stringbuilder("abc")
```



```java
public static void allSubstrings(String str){
    for(int i = 0;i < str.length();i++){
        for(int j = i; j < str.length();j++){
            // System.out.print("(" + i + ", " + (j + 1) + ") ");
            System.out.println(str.substring(i, j + 1));
        }
        System.out.println();
    }
}
```

2//

string str = "abcde"

2// String str = "abaqqbcde7fghiiiigh" ← user

'abACA127A'

char ch = 'a'; ← user

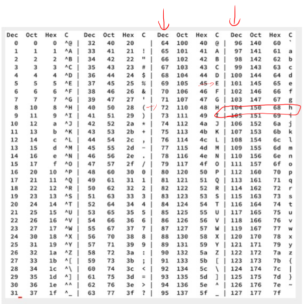
return count of ch

int count = ~~0~~ ~~1~~ ~~2~~ 34

3) "abAACCaapqraRSqqr"

'a' ≤ ch ≤ 'z'

'A' ≤ ch ≤ 'Z'



Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C
0	0	0	^@	32	40	20		64	100	40	@	96	140	60	`
1	1	1	^A	33	41	21	!	65	101	41	A	97	141	61	a
2	2	2	^B	34	42	22	"	66	102	42	B	98	142	62	b
3	3	3	^C	35	43	23	#	67	103	43	C	99	143	63	c
4	4	4	^D	36	44	24	$	68	104	44	D	100	144	64	d
5	5	5	^E	37	45	25	%	69	105	45	E	101	145	65	e
6	6	6	^F	38	46	26	&	70	106	46	F	102	146	66	f
7	7	7	^G	39	47	27	'	71	107	47	G	103	147	67	g
8	10	8	^H	40	50	28	(	72	110	48	H	104	150	68	h
9	11	9	^I	41	51	29	)	73	111	49	I	105	151	69	i
10	12	a	^J	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	b	^K	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	c	^L	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	d	^M	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	e	^N	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	f	^O	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	^P	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	^Q	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	^R	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	^S	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	^T	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	^U	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	^V	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	^W	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	^X	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	^Y	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	^Z	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	^[	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	^\	60	74	3c	<	92	134	5c	\	124	174	7c	|
29	35	1d	^]	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	^^	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	^_	63	77	3f	?	95	137	5f	_	127	177	7f	

— Figure 1. ASCII table output, including decimal, octal, hex, and character values.

P = 80
A = 65

p = 112
a = 97

→ ch - 'a' = CH - 'A'

(10) → CH = (char)(ch - 'a' + 'A')   capital

(10) → ch = (char)(CH - 'A' + 'a')   small

? - 'a' = 'H' - 'A'

? = 72 - 65 + 97

= 7 + 97

= 104

= 'h'

```
public static void stringConversion(String str) {
                    String newString = "";

                    for(i = 0 ; i<str.length() ; i++) {
                              char iIndexPrCh = str.charAt(i);
                              if(iIndexPrCh >='a' && iIndexPrCh <='z'){

                                        int diffCharWithSmallA = iIndexPrCh - 'a';
                                        int a = 'A';
                                        char newCharUpper = (char) (a + diffCharWithSmallA);
                                        newString += newCharUpper ;
                              }

                                        if(iIndexPrCh >='A' && iIndexPrCh <='Z){

                                        int diffCharWithUpperA = iIndexPrCh - 'A';
                                        int b = 'a';
                                        char newCharLower = (char) (b + diffCharWithUpperA);
                                        newString += newCharLower ;
                               }
                    }
          System.out.println(newString);
          }
```