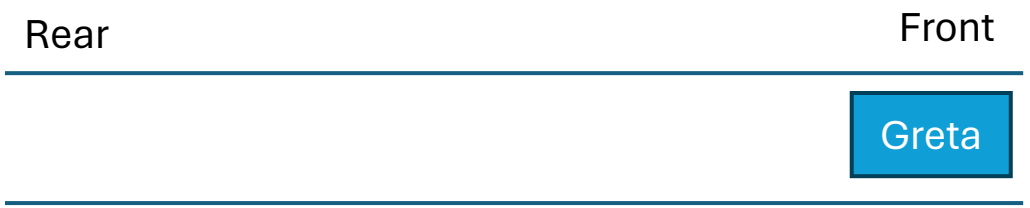
Queue

a. ICA öppnar och kön till kassan är tom

b. Kalle ställer sig i kön

c. Greta ställer sig i kön

d. Kalle blir expedierad och lämnar kön



e. Stina ställer sig i kön

f . Greta blir expedierad och lämnar kön

g. Olle ställer sig i kön

b.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
+Kalle
----------Queue----------
1: Kalle
--------------------------
```

c.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
+Greta
----------Queue----------
1: Kalle
2: Greta
--------------------------
```

d.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
-
----------Queue----------
1: Greta
--------------------------
```

e.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
+Stina
----------Queue----------
1: Greta
2: Stina
--------------------------
```

f.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
-
----------Queue----------
1: Stina
--------------------------
```

g.

```
0: Exit
+: To enqueue a person (+ or +Adam)
-: to deueue a person from queue
+Olle
----------Queue----------
1: Stina
2: Olle
--------------------------
```

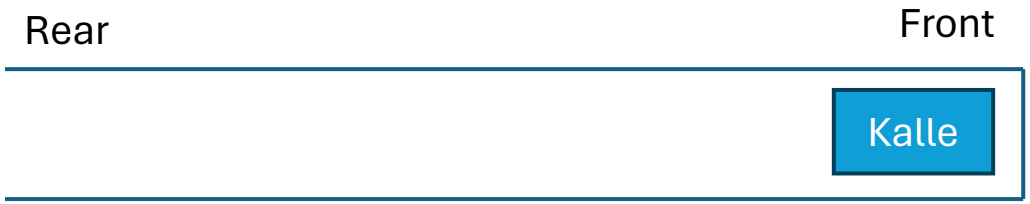
Stack

a. ICA öppnar och stacken till kassan är tom

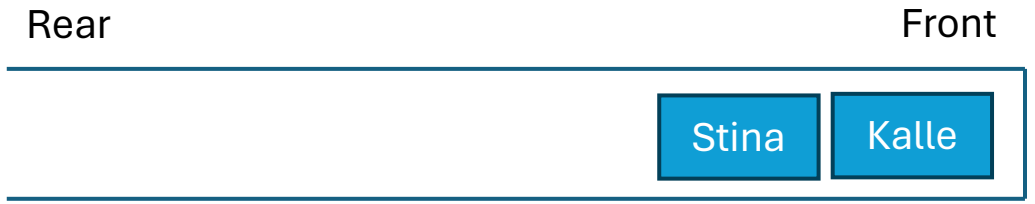
b. Kalle ställer sig i stacken

c. Greta ställer sig i stacken

d. Greta blir expedierad och lämnar kön



e. Stina ställer sig i Stacken



f . Stina blir expedierad och lämnar stacken

g. Olle ställer sig i stacken

Övning 4:

0	1	2	3	4	5	6	7	8	9
(	[	{	}	]	(	{	}	)	)

Stack: Adds only the opening parameters

0.			(	
1.		[	(	
2.	{	[	(	
3.		[	(	Index 3 = }, check if the top of stack matches with {, if so then pop the top
4.			(	Ind =], checks if the top matches with [,, if so then pop the top
5.		(	(	
6.	{	(	(	
7.		(	(	Index 7 = }, check if the top is {, if so, then remove it
8.			(	Index 8 =), check if the top is (, if so, then remove it
9.				Index 9 =), check if the top is (, if so, then remove it, since the stack is empty then return true that the format is correct

Övning 4:

0	1	2	3
(	{	)	}

Stack: Adds only the opening parameters

0.		(	
1.	{	(	
2.	{	(	Ind 3 =), check if the top matches (, since not so it return false and end the program

Rescursive

```
static int RecursiveOdd(int n)
{
    if (n == 1)
    {
        return 1;
    }
    return (RecursiveOdd(n - 1) + 2);
}
```

RecursiveOdd(1):

n = 1: => returns 1

RecursiveOdd(3):

N = 3:
Calls RecursiveOdd(3-1)+2 => RecursiveOdd(2)+2 => (RecursiveOdd(2-1)+2) +2 =>
(RecursiveOdd(1)+2) +2 => 1+2+2 = 5

N = 5:
RecursiveOdd(5-1)+2 => RecursiveOdd(4)+2 => (RecursiveOdd(3)+2) +2 => (RecursiveOdd(2)+2) + 4 =>
(RecursiveOdd(1)+2) + 6 => 1+2+6 = 9