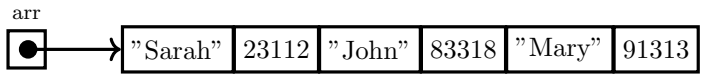
Exercise Sheet, Week 2

1 Arrays and linked lists

Treat memory as a large array `Mem` in this exercise.

1. A list of `n` student entries is stored as an array and the first location of the array is stored in the variable `arr`. Each student entry consists of a name (stored as a string) followed by an ID (stored as an integer); each of those occupies one location. Example:



   Write a program (pseudocode) to find the student with ID equal to 11111 and print his/her name. If such student is not in the array, throw an exception.

```
String find() {
}
```

2. A list of students is stored as a linked list with the location of the first entry stored in `first` and the location of the last entry stored in `last`. Example:

   Write a program (pseudocode) which attaches student with name `"Peter"` and ID `11111` at the end of the list.

```
newblock = allocate_memory(3);
;
```

2 Stacks stored in arrays

Recall from the lecture:

```
// Initialize an empty stack:
stack = new int[MAXSTACK];
stack_size = 0;
```

Write pseudocode for `pop` and `push`. Make sure that you raise `EmptyStackException` whenever you `pop` from an empty stack and that you raise `OutOfMemoryException` whenever you `push` to a full stack.

```
int pop() {
```

```
4
5
6
7 }
```

```
1 void push(int x) {
2
3
4
5
6
7 }
```

3 Mod and div

Recall that, for numbers a and b such that $b > 0$, a div b is the result of dividing a by b and discarding the remainder, and a mod b is the reminder. Moreover, $(a \text{ div } b) * b + a \bmod b = a$ and that $0 \leq a \bmod b < b$.

Calculate the following:

1. $20 \text{ div } 3 = ?$ $\quad 20 \bmod 3 = ?$
2. $21 \text{ div } 7 = ?$ $\quad 21 \bmod 7 = ?$
3. $-25 \text{ div } 4 = ?$ $\quad -25 \bmod 4 = ?$
4. $-33 \text{ div } 11 = ?$ $\quad -33 \bmod 11 = ?$

Answers:

4 (Circular) Queues stored in arrays

Recall from the lecture:

```
1 // Initialize an empty queue:
2 queue = new int[MAXQUEUE];
3 rear  = 0;
4 front = 0;
```

Write pseudocode for `dequeue`. Make sure you rise `EmptyQueueException` whenever you `dequeue` from an empty queue.

```
1 int dequeue () {
2
3
4
5
6
7
8 }
```

(Recall that the queue is empty whenever `rear == front`.)