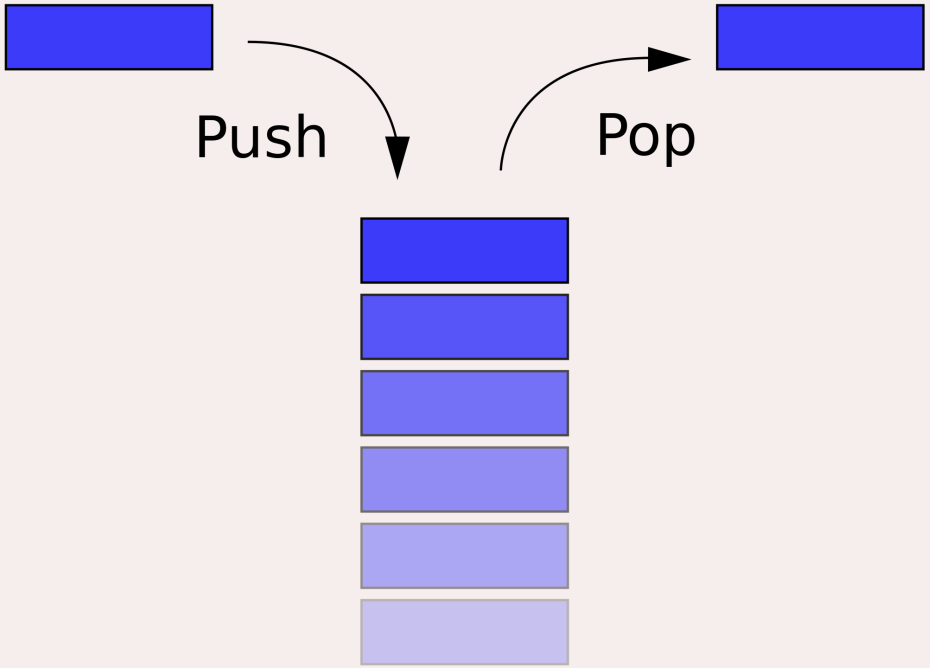
COMP20003 Workshop Week 5

1	Stacks
2	Queue
3	Assignment 1 Q&A
4	Lab: • Implement a stack (based on a linked list), or • working on your project

ADT: Stack (LIFO)

http://www.123rf.com/stock-photo/tyre.html



https://simple.wikipedia.org/wiki/Stack_(data_structure)

Stack Operations

`create:` create a new, empty stack

`push:` add an element into stack

`pop:` remove an element from stack

`isEmpty:` check if stack is empty

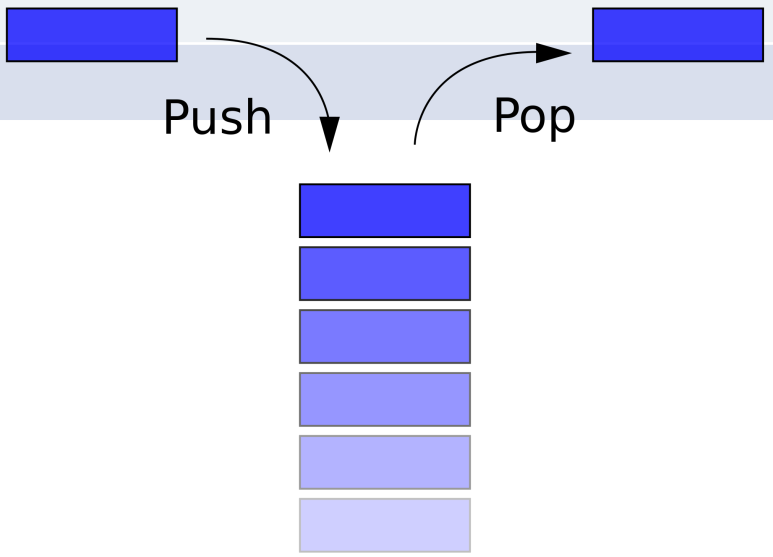
`delete:` free associated memory

Q5.1: Stack Implementation

We need:

- A structure for keeping the stack's elements, and
- A variable that keeps track of the stack's head.

Structure	Head variable= ?
Array	
Linked list	
???	



Stacks: (Dynamic) Array Implementation

```
#define INIT_SIZE 8
typedef struct {
  int a[???];       // array of int
  int head;     // index of stack head
} stack_adt;

// stack_adt createStack() is also ok
void createStack( stack_adt *ps) {
  ps->head= -1; // -1 or 0 ?
}
```

Stacks: (Dynamic) Array Implementation

```
#define INIT_SIZE 8
typedef struct {
   int *a;       // array of int
   int head;     // index of stack head
   int size;     // max size of array
} stack_adt;

// stack_adt createStack() is also ok
void createStack( stack_adt *ps) {
  ps= malloc(sizeof(*ps));   // ?

  ps->size= INIT_SIZE;


  ps->head= -1; // ps->a[ ps->head ] is the stack head
}
```

Stacks: (Dynamic) Array Implementation

```
#define INIT_SIZE 8
typedef struct {
   int *a;       // array of int
   int head;     // index of stack head
   int size;     // max size of array
} stack_adt;

// stack_adt createStack() is also ok
void createStack( stack_adt *ps) {
  ps= malloc(sizeof(*ps));   // ?
  assert(ps);
  ps->size= INIT_SIZE;
  ps->a= malloc( ps->size * sizeof(*(ps->a)) );
  assert( ps->a );
  ps->head= -1; // // ps->a[ ps->head ] is the stack head
}
```

Stacks: Array Implementation

```
typedef struct {int *a; int head, size;} stack_adt;

stack_adt push( int x )                ???

void push( stack_adt *ps, int x) {
      ps->a[ ++ps->head ]= x;                ???
}

int pop( stack_adt *ps ) {
   return ps->a[ ps->head-- ];                  ???
}

void deleteStack( stack_t *ps )  {   ???  }
```

Stacks: Array Implementation

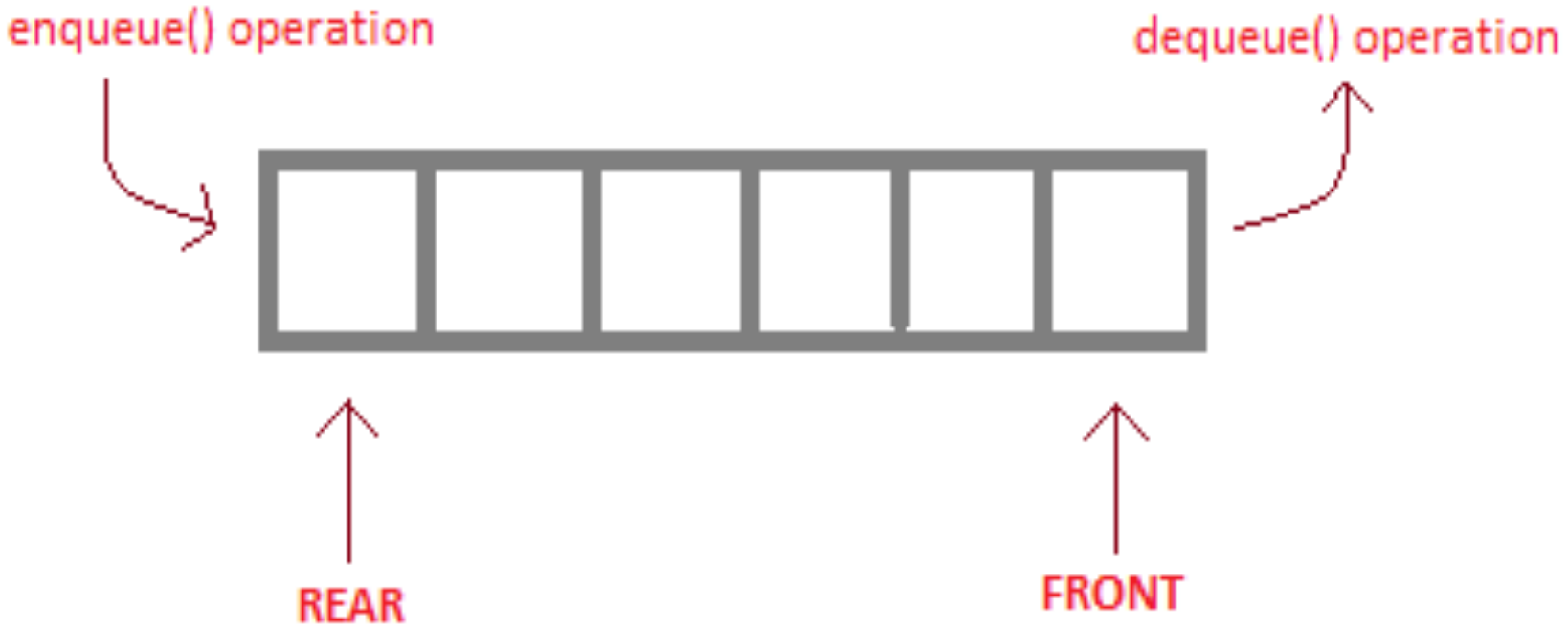
```
typedef struct {int *a; int head, size;} stack_adt;


void push( stack_adt *ps, int x) {
  if ( ps->head == ps->size - 1 ) {
    ps->size *= 2;
    ps->a= realloc( ps->size * sizeof(*(ps->a)) );
    assert(ps->a);
  }
  ps->a[ ++ps->head ]= x;
}

int pop( stack_adt *ps ) {
  assert( ps->head >= 0 );
  return ps->a[ ps->head-- ];
}
```

Stack: linked list implementation

It's straight-forward:

Note: In this case stack is a special linked list where we can efficiently insert or delete elements (but only at one end).

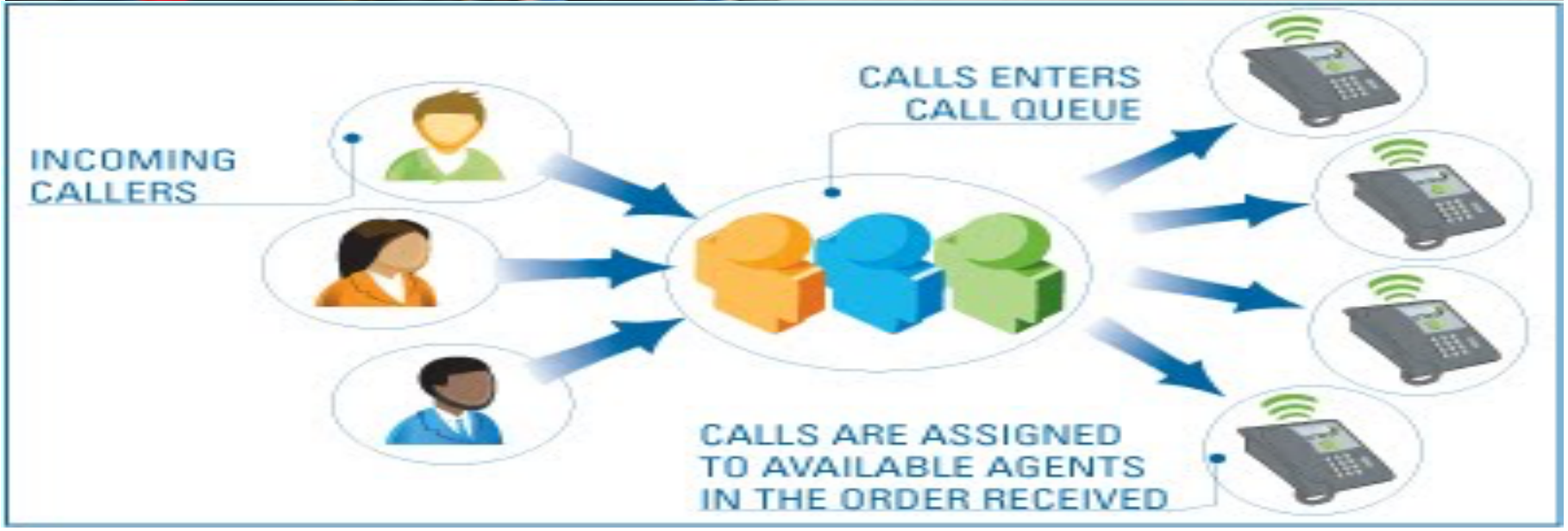
ADT: Queue (FIFO)



enqueue( ) is the operation for adding an element into Queue.

dequeue( ) is the operation for removing an element from Queue .

Data structure: Queue (FIFO)



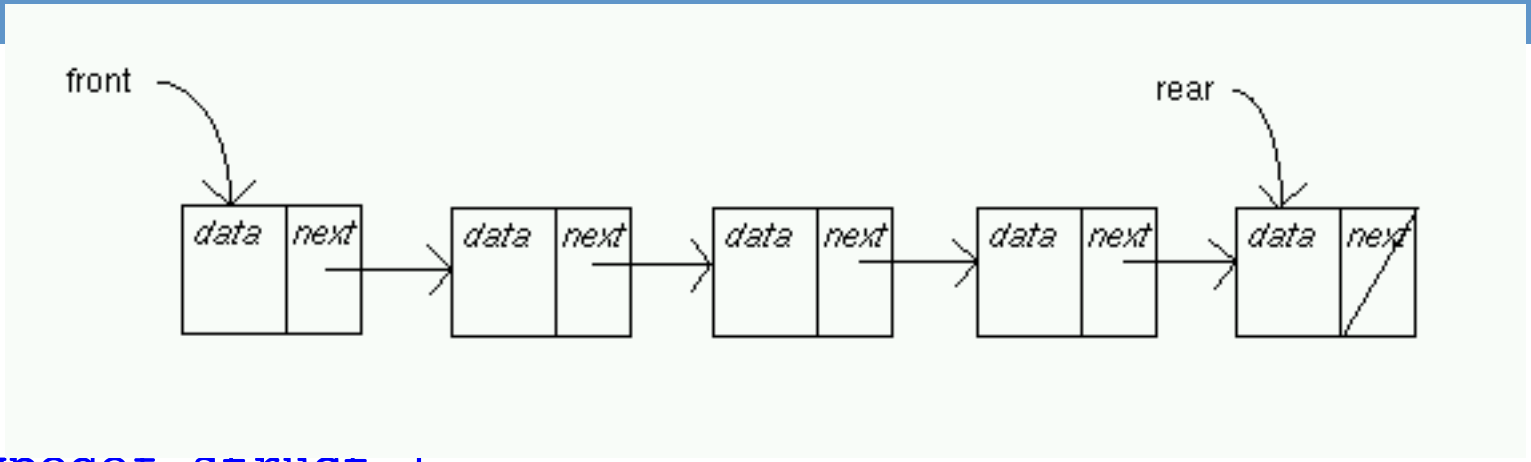
Queue Implementation

Using arrays?

Using linked list?

Any problem with arrays? with lists?

Queues & Linked Lists



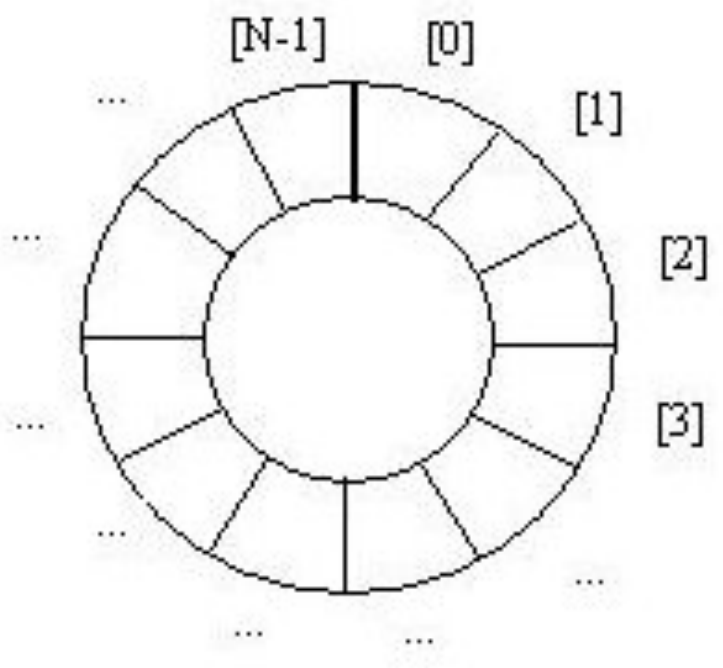

```
typedef struct {
    list_t front;   // example of list_t is
    list_t rear;    // typedef struct node * list_t
} queue_t;

void createQueue( queue_t *pq) {
    pq= malloc(sizeof(*pq));
    assert(pq);
    pq->front= pq->rear= NULL;
}
```

array implementation?

Queue Implementation

Using circular arrays?

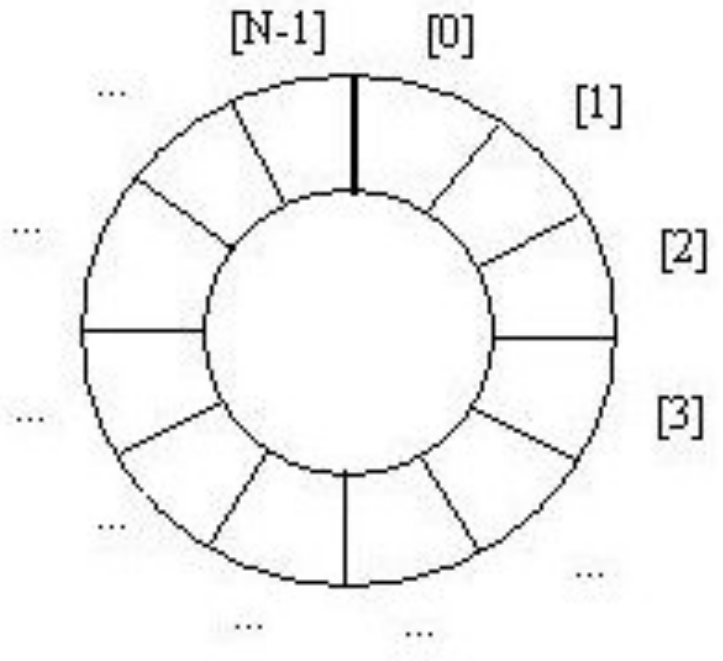
Circular arrays: 1D arrays, where the first element is considered as next to the last element. Hence, the element next to `a[i]` is `a[(i+1) mod N]`



Queue Implementation

Potential problems with circular arrays:

- static circular arrays?
- dynamic circular arrays?



Notes

For stacks and queues:

- For stacks and queues: implementation using linked lists is simpler and "more natural" than using arrays

For linked lists:

- A stack based on linked list is actually simpler and easier to program than a general linked list, and the same for queue…
- In many cases, when we want to use a linked list, remember that it could be just a stack or a queue

Assignment 1: ?