Activity #4: Stacks and Queues
Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 2, Question #5

b) Model 3, Question #7

c) Model 5, Question #12

d) Model 6, Question #14

Activity #4: Stacks and Queues

We are studying **data structures** - ways to organize and structure data. Implementing a data structure has pedagogical value, but as good (lazy?) computer scientists, we don't want to implement a new version of a familiar data structure for every situation. Instead, we want to design an **abstract data type (ADT)** that captures the key ideas of a data structure and can be used in many different situations.
This activity will give you some experience analyzing requirements and designing ADTs. In particular, you will investigate **stacks** and **queues**.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain a stack
- Identify the operations needed to support a stack
- Explain a queue
- Identify the operations needed to support a queue

Process Skill Goals

During the activity, students should make progress toward:

- Defining the operation function signatures for a stack
- Defining the operation function signatures for a queue

Model 1 Stacks - Real World

Consider the following situations:

- A pile of papers or books on a table, where the top item is usually the most recently added.
- In arithmetic expressions, each ")" matches the most recent unmatched "(". In Java programming, each "}" matches the most recent unmatched "{". In HTML, each </DIV> matches the most recent unmatched <DIV>. In each case, a parsing program must keep track of all unmatched symbols.
- If method A calls B, and B calls C, then when C ends control returns to B, and when B ends control returns to A. (Control returns to the most recently called method).
- As people, if we have too many tasks or questions of similar importance, we tend to focus on the most recent ones.

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

1. These situations are all examples of **stacks**. Summarize their key common characteristics in 2-4 complete English sentences.

2. List any variations or exceptions to these characteristics.

3. Trace the following actions in a stack (steps a through e are provided as examples).

	action		**contents**					**outcome**
a.	empty stack							
b.	get top item							ERROR - empty stack
c.	add A		A					
d.	add B		A	B				
e.	remove top item		A					B is removed
f.	get top item							
g.	add C							
h.	add D							
i.	get top item							
j.	add E							
k.	add F							

Otherwise, my whole career has just been flinging myself
at whatever is most overdue first and letting everything else stack up.
– Cathy Guisewite, American cartoonist, 1950-

Model 2 Stacks - Abstract Data Type

Note that stacks are often described as **Last-In, First-Out (LIFO)** or **First-In, Last-Out (FILO)** (particularly if you are Greek :-)).

Questions (5 min) Start time:

4. Based on the key characteristics of stacks that you identified above, **list at least 3 key operations** for a stack ADT, and **rank them** (last column) by **importance** (1=high, 5=low).

	action or operation	**rank**
a.		
b.		
c.		
d.		

5. Start with the most important stack operation above, and define a **method signature** including an appropriate name, input parameters, and return types. You may write the method below, or create a class file in your IDE. Use "T" as a generic placeholder for the type of object stored in the stack.

```
template<typename T>
class Stack {
public:
    Stack(int maxSize);  // constructor

    ?

    ?

    ?

    ?

    ?

};  // end class Stack
```

Review progress with the facilitator before continuing.

Model 3 Stacks - Array Implementation

A stack ADT can be implemented in multiple ways, as you will see. For now, we will consider how to implement a stack using an array, e.g.:

```
template<typename T>
class Stack {
private:
    T* data;       // array to store stack elements
    int capacity; // maximum size of stack
    int size;     // current number of elements

public:
    Stack(int maxSize) {
        this->capacity = maxSize;
        this->size = 0;
        this->data = new T[maxSize];
    }

    // methods
};

```

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (10 min)   Start time:

6. **Show the contents of the data array after each operation.** Assume we start with an empty stack. If you use additional fields (e.g. to keep track of positions in the array), add a column for each one and show how its value changes.

	Operation	[0]	[1]	[2]	[3]			
a.	Create new data structure.							
b.	Add value 'A'.							
c.	Add value 'B'.							
d.	Remove value.							
e.	Add value 'C'.							
f.	Get current size.							
g.	Remove value.							
h.	Add value 'D'.							

7. For each stack method, write a complete English sentence to describe **how it could be implemented** using an array, and its **O() performance**.
Hint: Ideally, the methods should be O(1), or O(N) at worst.

	method	**implementation**	**O()**
a.			
b.			
c.			
d.			
e.			

Review progress with the facilitator before continuing.

Model 4   Queues - Real World

Consider the following situations:

a) People standing in a checkout line at a store or fast-food restaurant.

b) Suitcases or packages on a conveyor belt.

c) Phone callers on hold waiting for the "next available customer service representative".

d) Documents waiting to be printed on a printer.

Refer to Model 4 above as your team develops consensus answers to the questions below.

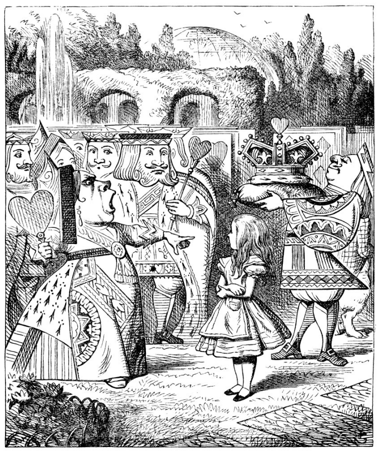
Questions (10 min)   Start time:

8. These situations are all examples of **queues**. Summarize their key common characteristics in 2-4 complete English sentences.

9. List any variations or exceptions to these characteristics.

10. (3 min) Trace the following actions in a queue (steps a through e are provided as examples).

	action		**contents**					**outcome**
a.	empty queue							
b.	get first item							ERROR - empty queue
c.	add A		A					
d.	add B		A	B				
e.	remove first item		B					A is removed
f.	get first item							
g.	add C							
h.	add D							
i.	get first item							
j.	add E							
k.	add F							



The Queen had only one way of settling all difficulties, great or small.
'Off with his head!' she said, without even looking round.
— Lewis Carroll, Alice in Wonderland

Model 5 Queues - Abstract Data Type

Note that queues are often described as **First-In, First-Out (FIFO)** or **Last-In, Last-Out (LILO)** (particularly if you are lazy :-)).

Questions (5 min) Start time:

11. Based on the key characteristics of queues that you identified above, **list at least 3 key operations** for a queue ADT, and **rank them** (last column) by **importance** (1=high, 5=low). Review progress with the facilitator before continuing.

	action or operation	**rank**
a.		
b.		
c.		
d.		

12. Start with the most important queue operation above, and define a method signature including an appropriate name, input parameters, and return types. You may write the method below, or create a class file in your IDE. Use "T" as a generic placeholder for the type of object stored in the queue.

```
template<typename T>
class Queue {
public:
    Queue(int maxSize);  // constructor

    ?

    ?

    ?

    ?

    ?

};  // end class Queue

```

Review progress with the facilitator before continuing.

Model 6 Queues - Array Implementation

A queue ADT can be implemented in multiple ways, as you will see. For now, we will consider how to implement a queue using an array, e.g.:

```
template<typename T>
class Queue {
private:
    T* data;       // array to store queue elements
    int capacity; // maximum size of queue
    int start;    // index of first element
    int end;      // index after last element

public:
    Queue(int maxSize) {
        this->capacity = maxSize;
        this->start = 0;
        this->end = 0;
        this->data = new T[maxSize];
    }

    // methods
};

```

Refer to Model 6 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

13. Show the contents of the data array after each operation. Assume we start with an empty queue. If you use additional fields (e.g. to keep track of positions in the array), add a column for each one and show how its value changes.

	Operation	**[0]**	**[1]**	**[2]**	**[3]**			
a.	Create new data structure.							
b.	Add value 'A'.							
c.	Add value 'B'.							
d.	Remove value.							
e.	Add value 'C'.							
f.	Get current size.							
g.	Remove value.							
h.	Add value 'D'.							

14. For each queue method, write a complete English sentence to describe **how it could be implemented** using an array, and its **O() performance**.
Hint: Ideally, the methods should be O(1), or O(N) at worst.

	method	**implementation**	**O()**
a.			
b.			
c.			
d.			
e.			

15. For an array implementation, a queue is somewhat more difficult than a stack. Explain why this is the case and what extra thought is required to implement a queue.

Review progress with the facilitator before continuing.

An Englishman, even if he is alone, forms an orderly queue of one.

– George Mikes, British writer, 1912-1987