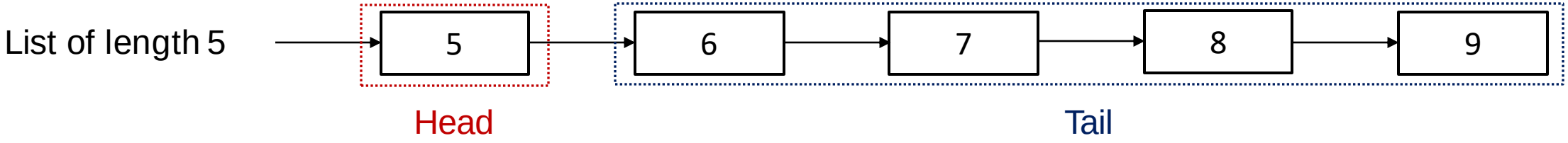
CSCI2100C Data Structures
Tutorial 05 – List

LI Muzhi 李木之
mzli@cse.cuhk.edu.hk

Slides adapted from Professor Leung Ho-Fung's Lecture notes
Icons made by https://freepik.com

Recall: List

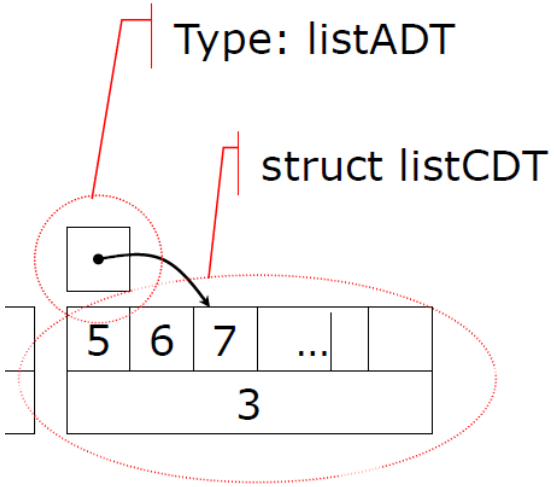



```c
typedef struct listCDT *listADT;

typedef int listElementT;

listADT EmptyList(void);
listADT Cons(listElementT, listADT);
listElementT Head(listADT);
listADT Tail(listADT);
int ListIsEmpty(listADT);
```

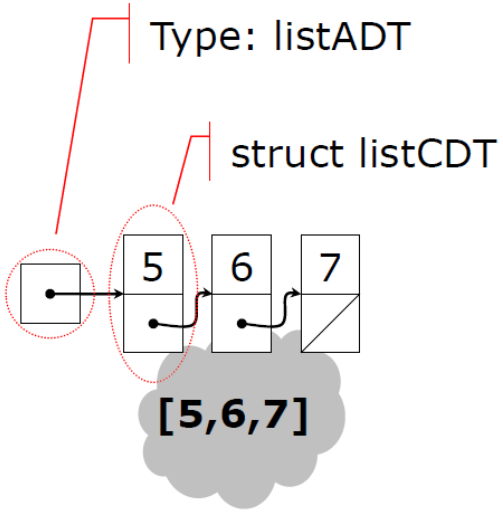
list.h

```c
listADT Append(listADT, listElementT);
int ListLength(listADT);
```

Implemented using functions in list.h



List version 1.0
Implemented by Array



List version 2.0
Implemented by Linked List

Exercise 1 – Print List

Write a program in **application file** that print the contents of a list. Use the ListADT to complete the exercise. Use the following function prototype.

```
#include "list.h”
void printList(listADT L1);
```

Example:

printList([]): “Empty List”

printList([2, 3, 4]): “2 3 4”

Exercise 1 – Example

List of length 5

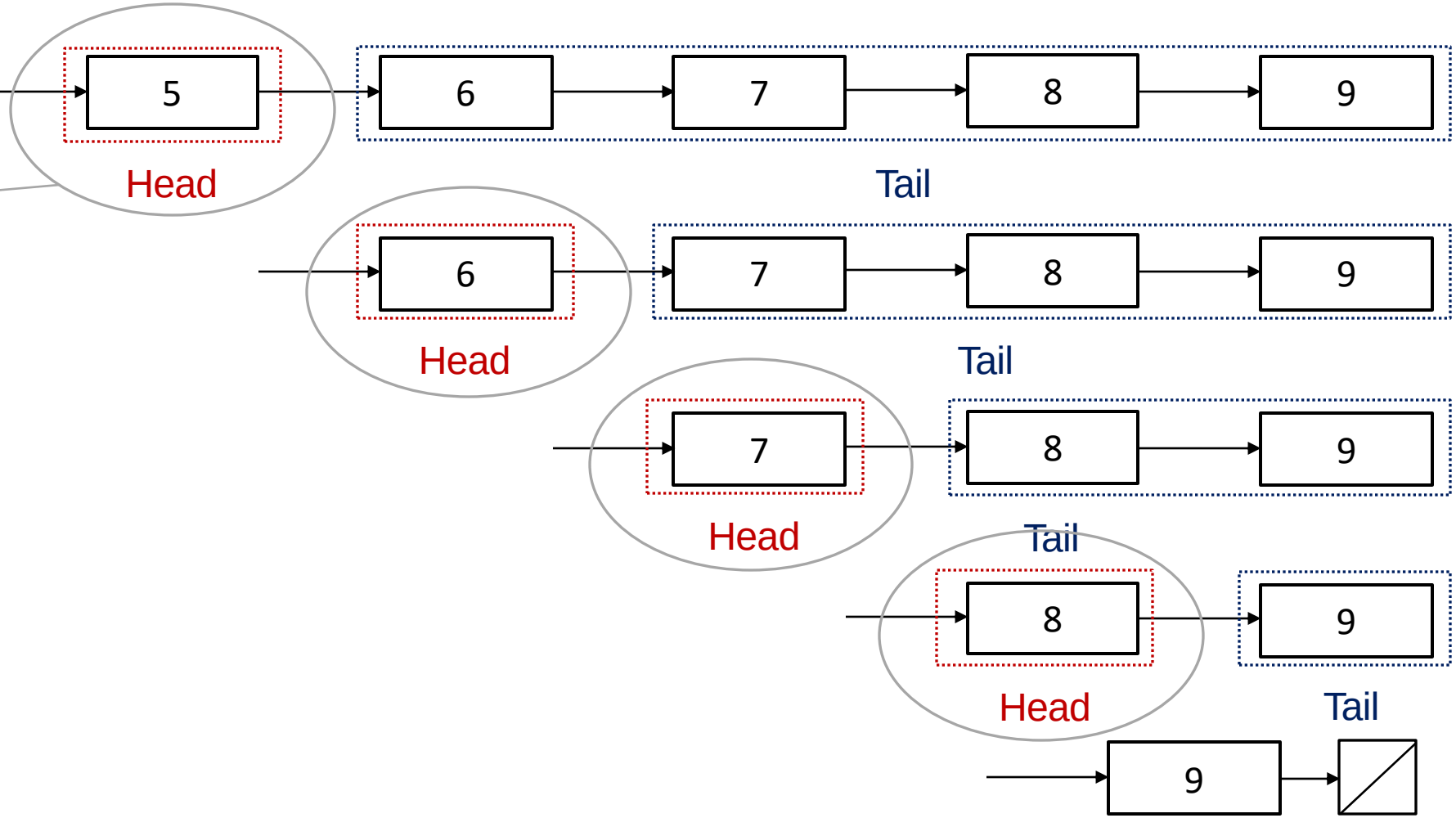
Output: 5

Output: 5 6

Output: 5 6 7

Output: 5 6 7 8

Output: 5 6 7 8 9



Exercise 1 – Answer

```
void printList(listADT l1) {
    if (ListIsEmpty(l1)) {
        printf("List is now empty.\n");
    }
    else {
        printf("%d ", Head(l1));
        printList(Tail(l1));
    }
}
```

Exercise 2a – List Concatenation

Write a program **application file** that concatenates two lists of integers. Use the List ADT to complete the exercise. Use the following function prototype.

```
#include "list.h“
listADT Append(ListADT, ListElementT);  // Suppose that Append function is implemented
listADT Concat(listADT L1, listADT L2);
```

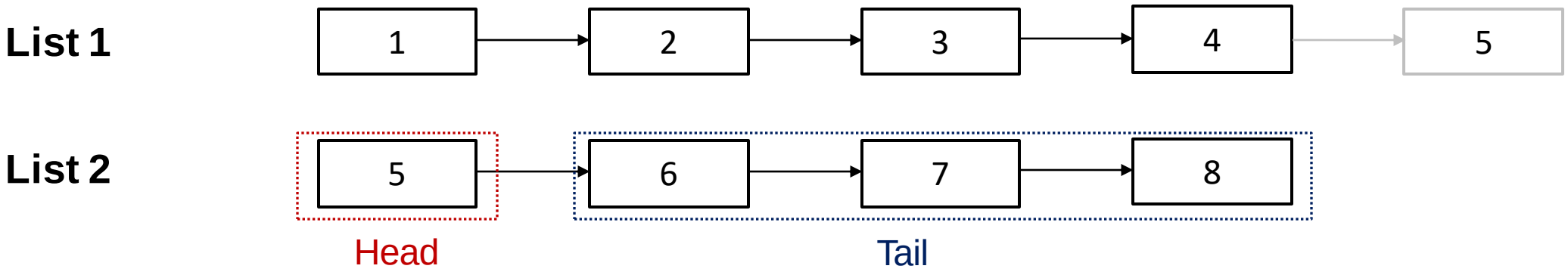
Example:

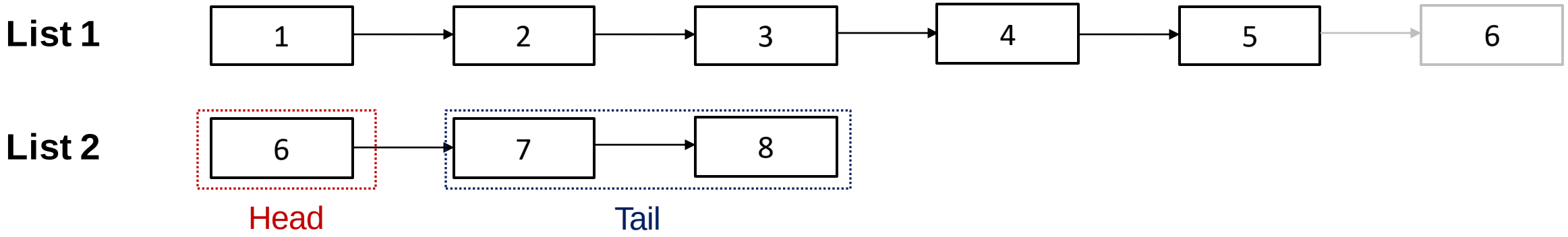
Concat([], []) = []

Concat([], [2, 3]) = [2, 3]

Concat([1, 2], [2, 3]) = [1, 2, 2, 3]

Exercise 2a – Explanation





Exercise 2a – Answer

```
listADT Concat(listADT l1, listADT l2) {
    if (ListIsEmpty(l1)){
        return l2;
    }
    else if (ListIsEmpty(l2)) {
        return l1;
    }
    else {
        while(!ListIsEmpty(l2)){
            listElementT head = Head(l2);
            l2 = Tail(l2);
            l1 = Append(l1, head);
        }
        return l1;
    }
}
```

→

Or simply, this standard version

```
listADT Concat(list1, list2) {
    if (ListIsEmpty(list2)) return(list1);
    return (Concat(Append(list1, Head(list2)), Tail(list2)));
}
```

Exercise 2b – List Concatenation

Write a program **application file** that concatenates two lists of integers. Use the List ADT to complete the exercise. Use the following function prototype.

```
#include "list.h“
listADT Append(ListADT, ListElementT);  // Append function is not allowed to be used…
listADT Concat(listADT L1, listADT L2);
```

Example:

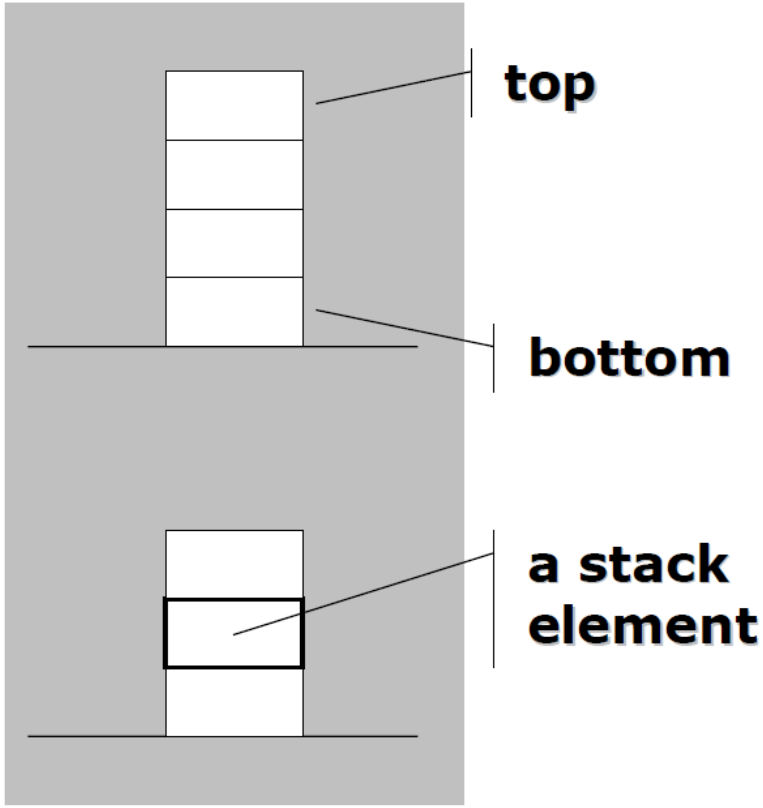
Concat([], []) = []
Concat([], [2, 3]) = [2, 3]
Concat([1, 2], [2, 3]) = [1, 2, 2, 3]

Recall: Stack

- A **stack** is a LIFO data type

```
typedef struct stackCDT *stackADT;
typedef int stackElementT;

stackADT EmptyStack(void);
void Push(stackADT stack, stackElementT element);
stackElementT Pop(stackADT stack);
int StackDepth(stackADT stack);
int StackIsEmpty(stackADT stack);
```

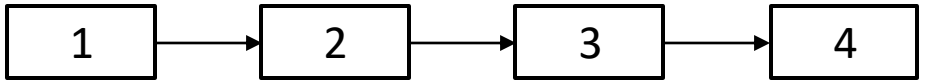
stack.h



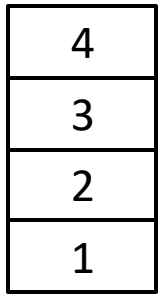
Exercise 2b – Answer

```
listADT Concat(listADT l1, listADT l2) {
    if (ListIsEmpty(l1)){
        return l2;
    }
    else if (ListIsEmpty(l2)) {
        return l1;
    }
    else {
        stackADT stack = EmptyStack();
        while(!ListIsEmpty(l1)){
            Push(stack, Head(l1));
            l1 = Tail(l1);
        }
        while(!StackIsEmpty(stack)){
            l2 = Cons(Pop(stack), l2);
        }
        return l2;
    }
}
```

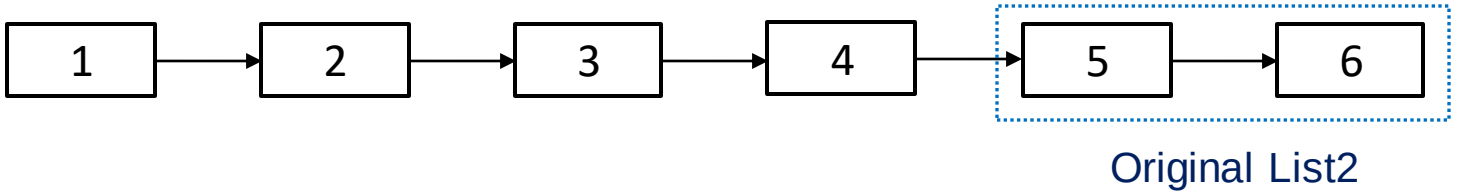
List 1



Create a stack based on list 1

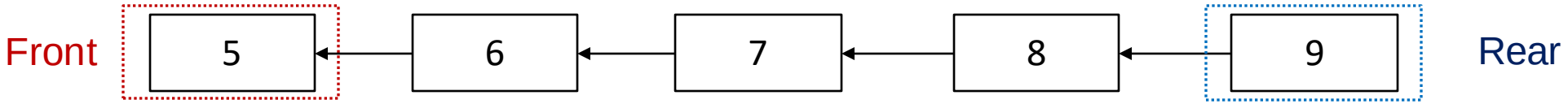


Construct a new list based on the stack and List2



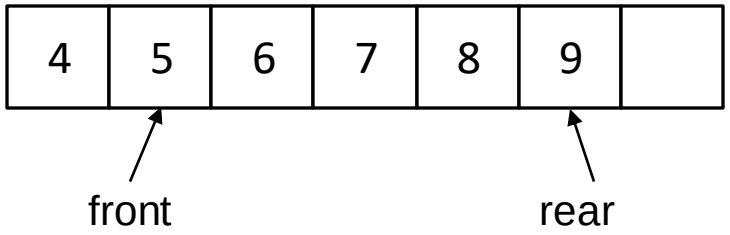
Recall: Queue

- A Queue is a FIFO data type

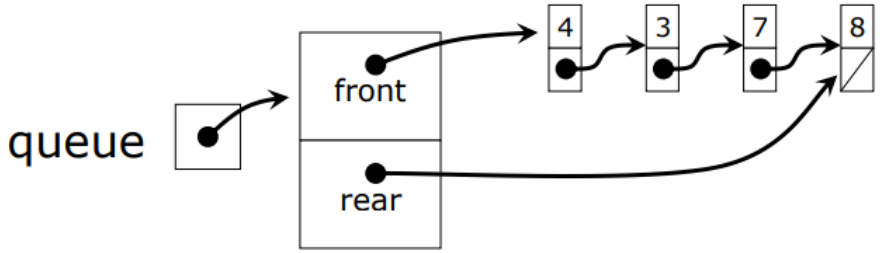



```
typedef struct queueCDT * queueADT;

typedef int queueElementT;

queueADT EmptyQueue(void);
void  Enqueue(queueADT, queueElementT);
queueElementT Dequeue(queueADT);
int QueueLength(queueADT);
int QueueIsEmpty(queueADT);
```

queue.h



Queue ver. 1.0 – implemented by array



Queue ver. 2.0 – implemented by Linked List

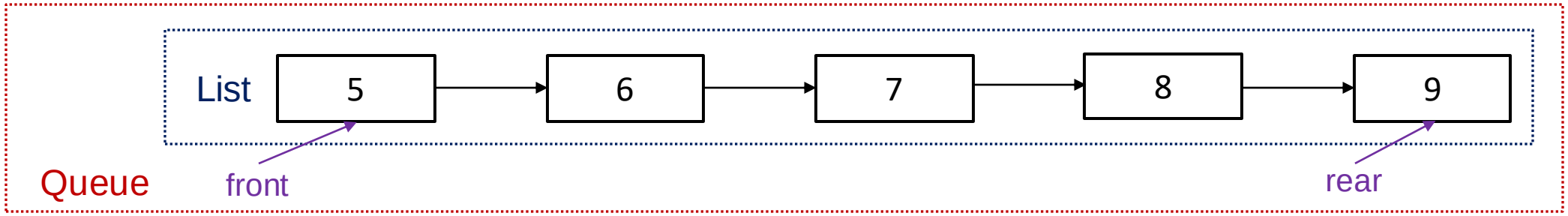
- Can we implement Abstract Data Type Queue by Abstract Data Type List?

YES!

- Programmers are “lazy” … 
- Code Reuse is a good common practice in Software development industry 😈
- Why? Reduce mistakes and save debugging time ⏱

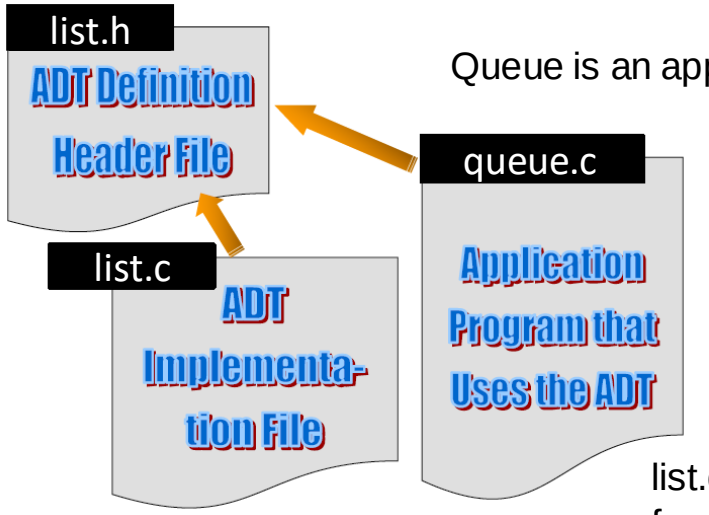
Task: “Queue ver. 3.0” – Implement Queue by List

- Implement your own **queue.c** file that **store (encapsulate) all queue elements in a list**.
- File queue.h, list.h and list.c are provided, but you don’t know the detail implementation of list ADT




```
typedef struct queueCDT * queueADT;

typedef int queueElementT;

queueADT EmptyQueue(void);
void  Enqueue(queueADT, queueElementT);
queueElementT Dequeue(queueADT);
int QueueLength(queueADT);
int QueueIsEmpty(queueADT);
```

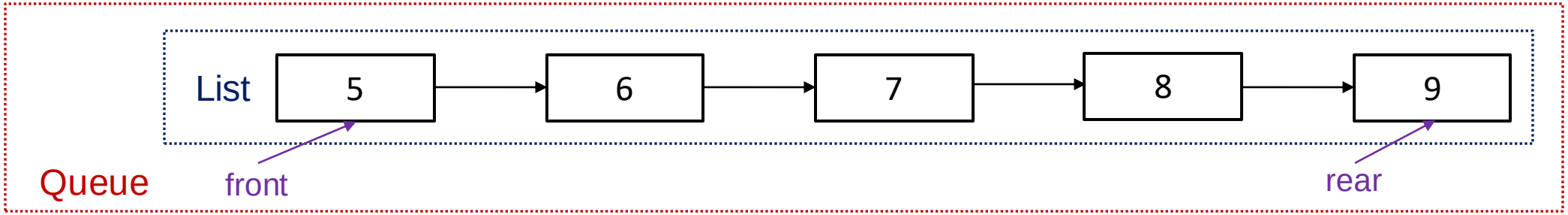
queue.h (you need to implement these functions in queue.c)



Queue is an application of ADT List

list.c is just like a black-box for queue.c

Answer – Data Representation

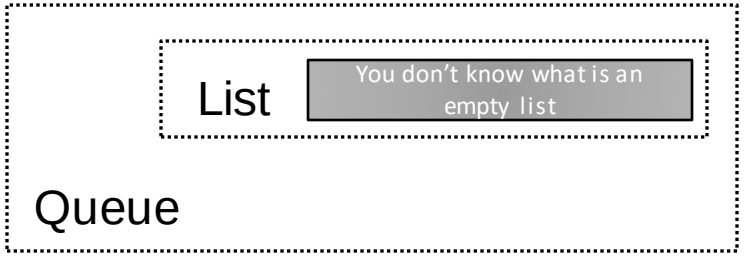



```c
typedef struct cellT {
    queueElementT value;
    struct cellT* next;
} cellT;

struct queueCDT{
    queueElementT *queue;
    cellT * front;
    cellT * rear;
};
```

Do we still need this part?

```c
struct queueCDT{
    listADT elements;
};
```

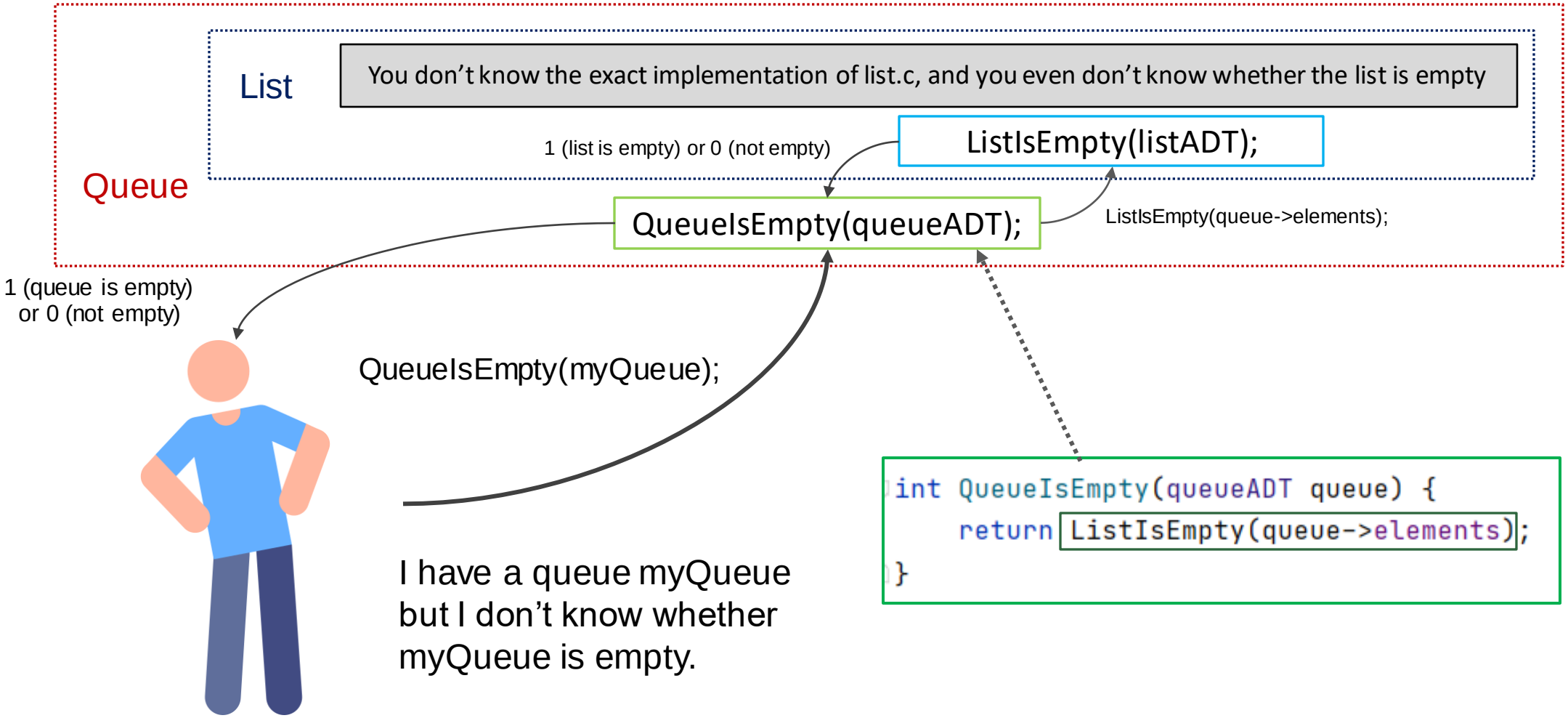
```c
queueADT EmptyQueue(void) {
    queueADT queue;
    queue = (queueADT) malloc(sizeof(*queue));
    queue->elements = EmptyList();
    return queue;
}
```



Code from Queue v2.0

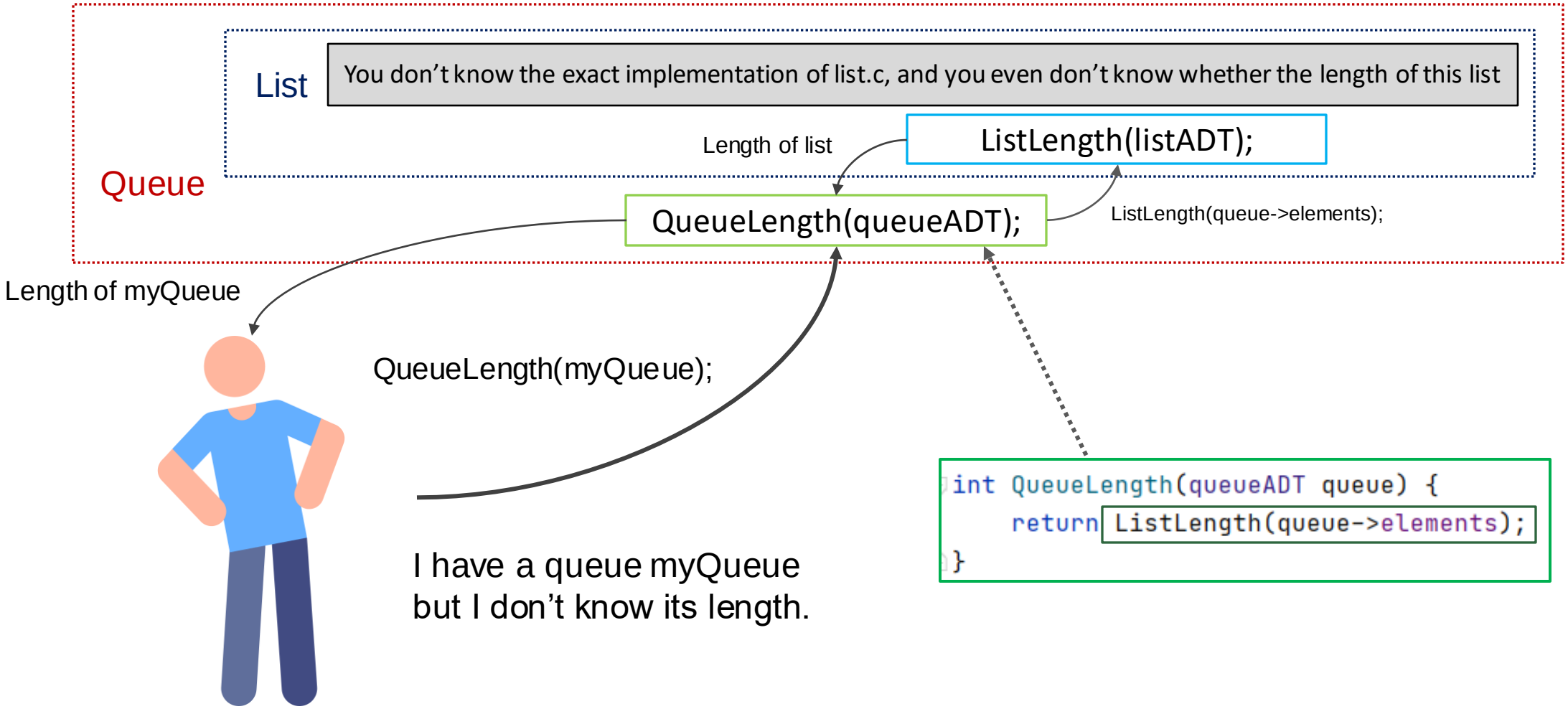
We just know the queue is empty, but we don't know what exactly implement an empty list.

Answer – QueueIsEmpty function




```c
int QueueIsEmpty(queueADT queue) {
    return ListIsEmpty(queue->elements);
}
```

Answer – QueueLength function




```
int QueueLength(queueADT queue) {
    return ListLength(queue->elements);
}
```

Answer – Enqueue function

Queue

List

You don't know the exact implementation of list.c

Append(listADT, listElementT);

Enqueue(queueADT, queueElementT);

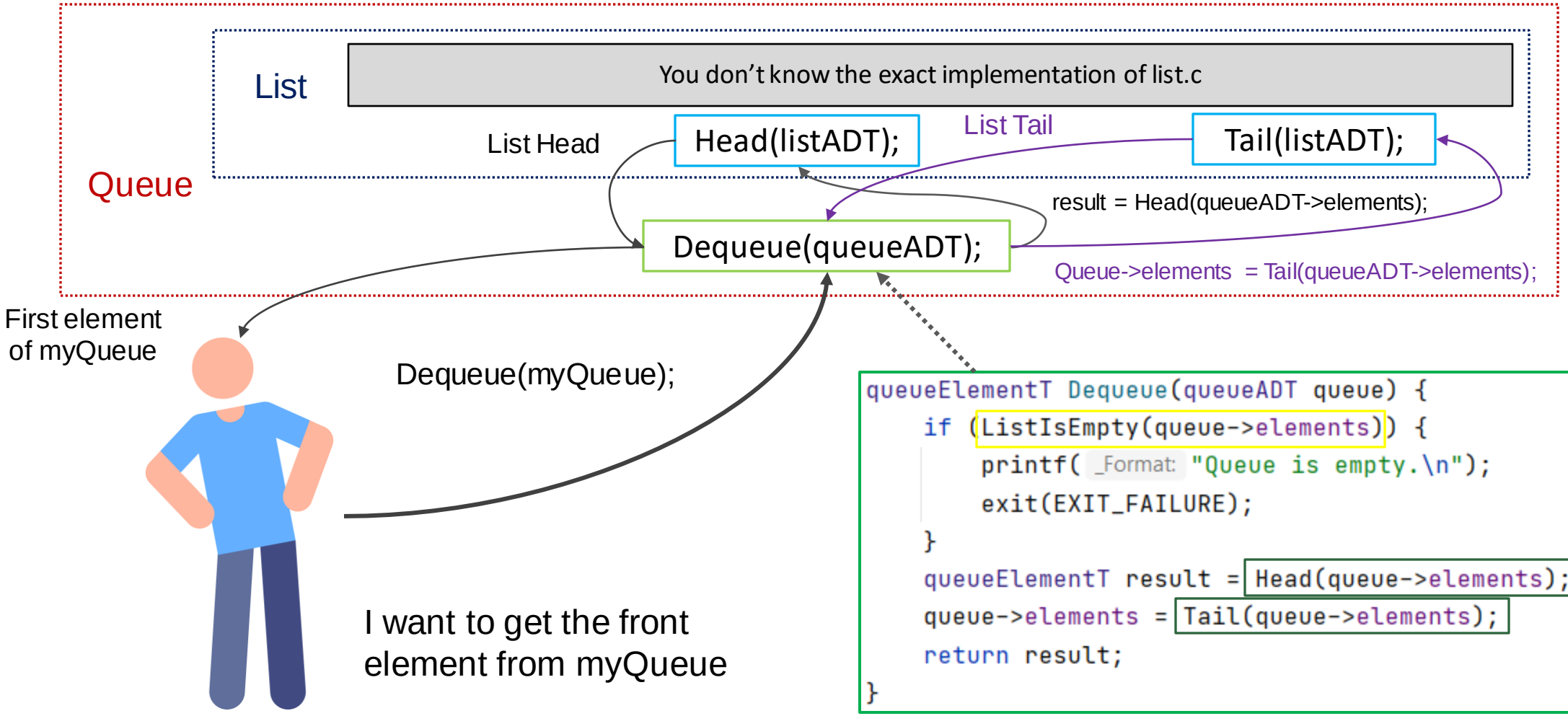
Append(queueADT->elements, queueElementT);

Enqueue(myQueue, x);

```
void Enqueue(queueADT queue, queueElementT element) {
    queue->elements = Append(queue->elements, element);
}
```

I want to enqueue x into myQueue.

Answer – Dequeue function




```
queueElementT Dequeue(queueADT queue) {
    if (ListIsEmpty(queue->elements)) {
        printf( _Format: "Queue is empty.\n");
        exit(EXIT_FAILURE);
    }
    queueElementT result = Head(queue->elements);
    queue->elements = Tail(queue->elements);
    return result;
}
```

Concrete Implementation of QueueADT ver. 3.0

```c
struct queueCDT{
    listADT elements;
};

queueADT EmptyQueue(void) {
    queueADT queue;
    queue = (queueADT) malloc(sizeof(*queue));
    queue->elements = EmptyList();
    return queue;
}
```

```c
int QueueIsEmpty(queueADT queue) {
    return ListIsEmpty(queue->elements);
}

int QueueLength(queueADT queue) {
    return ListLength(queue->elements);
}
```

```c
void Enqueue(queueADT queue, queueElementT element) {
    queue->elements = Append(queue->elements, element);
}

queueElementT Dequeue(queueADT queue) {
    if (ListIsEmpty(queue->elements)) {
        printf( _Format: "Queue is empty.\n");
        exit(EXIT_FAILURE);
    }
    queueElementT result = Head(queue->elements);
    queue->elements = Tail(queue->elements);
    return result;
}
```

Thanks