

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# Lists

TEAM INFDEV

Hogeschool Rotterdam  
Rotterdam, Netherlands

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# Introduction

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Lecture topics

- We now begin discussing specific, useful data structures
- These are already well known and understood
- Perfect for learning how a data structure is designed
- We begin with lists

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# Problem discussion

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Introduction

- So far we have been dealing with a single datum in every variable
- For example, integer 0 in variable `i`
- Sometimes we need to store multiple things in the same variable

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Examples

- All players
- All the employees of the company
- All the trucks on the road
- All the aliens in the spaceship
- All the alien spaceships in the fleet
- ...

# With variables?

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1 truck1 = Truck(...)  
2 truck2 = Truck(...)  
3 ...  
4 truck10 = Truck(...)
```

## Examples

# With variables?

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1 truck1 = Truck(...)
2 truck2 = Truck(...)
3 ...
4 truck10 = Truck(...)
```

## Examples

- Does this work?
- What if we have more or less than 10 trucks?



Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# General idea

## Introduction

- To solve this problem, we want to have all the data in a single variable
- The variable contains thus an unknown number of values
  - Might be empty
  - Might have only one element
  - Might have hundreds of elements
  - ...

## Description

- To solve the issue, we will define an open-ended data structure
- The list is built as a linear chain of **nodes**
- In the simplest implementation, each node has
  - a **value**
  - a reference to the **next** elements
- We never really know how many elements we have in the list until we follow all the references through
- A special case is the empty list, which has no element and no reference to the next elements

```

1 +---+---+   +---+---+   +---+---+   +---+---+
2 | 3 | ----->| 7 | ----->| 4 | ----->|   |   |
3 +---+---+   +---+---+   +---+---+   +---+---+

```

## Description

- Consider a list with elements 3, 7, and 4
- We need four nodes (the last is empty), all referencing the next

## Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**

## Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**  
`Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?**

## Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**  
`Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?**  
`Node(1,Node(2,Empty))`
- **An empty list would be?**

## Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**  
`Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?**  
`Node(1,Node(2,Empty))`
- **An empty list would be?** `Empty`
- ...



## Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?**

## Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?**

## Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with? `x.IsEmpty`**
  - **We can read print its first value with? `x.Value`**
  - **We can print its second value with?**

## Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?** `x.Value`
  - **We can print its second value with?** `x.Tail.Value`
  - **We can print its third value with?**

## Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?** `x.Value`
  - **We can print its second value with?** `x.Tail.Value`
  - **We can print its third value with?** `x.Tail.Tail.Value`
  - ...

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# Technical details

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Introduction

- How is this done in Python?
- We shall build two data structures that, together, make up arbitrary lists
- We begin with the blueprints

# The blueprint (**THIS IS NOT CODE!**)

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1  Abstraction Empty =  
2      IsEmpty, which is always true  
3  
4  Abstraction Node =  
5      IsEmpty, which is always true  
6      Value, which contains the datum of this element of the list  
7      Tail, which contains the remaining nodes of the list
```

## Introduction



# The blueprint (**THIS IS NOT CODE!**)

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1  Abstraction Empty =  
2      IsEmpty, which is always true  
3  
4  Abstraction Node =  
5      IsEmpty, which is always true  
6      Value, which contains the datum of this element of the list  
7      Tail, which contains the remaining nodes of the list
```

## Introduction

- How do we translate this to Python?

# The blueprint (**THIS IS NOT CODE!**)

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1  Abstraction Empty =  
2      IsEmpty, which is always true  
3  
4  Abstraction Node =  
5      IsEmpty, which is always true  
6      Value, which contains the datum of this element of the list  
7      Tail, which contains the remaining nodes of the list
```

## Introduction

- How do we translate this to Python?
- Each abstraction becomes a class
- Each field is assigned under `__init__` to self

# The actual code

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

```
1 class Empty:
2     def __init__(self):
3         self.IsEmpty = True
4     Empty = Empty()
5
6 class Node:
7     def __init__(self, value, tail):
8         self.IsEmpty = False
9         self.Value = value
10        self.Tail = tail
```

**Note: we are switching to Python 3!**

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Examples of list usage

- We now wish to build a list with our data structures
- We will build a list based on the input of the user
- User specifies how many, and which elements must go in the list

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC
1

H


```
1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v), l
```

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC
1

H


```

1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v, l)

```

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[ isEmpty $\mapsto$ True ]

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[ isEmpty $\mapsto$ True ]

```

1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v, l)

```

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[ isEmpty $\mapsto$ True ]

```

1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v, l)

```

S

PC	l	count	i	v
3	ref(1)	5	0	80085

H

0	1
[ isEmpty $\mapsto$ True ]	[ isEmpty $\mapsto$ False; Value $\mapsto$ 80085; Tail $\mapsto$ ref(0) ]



# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1
...	[ isEmpty $\mapsto$ False; Value $\mapsto$ 80085; Tail $\mapsto$ ref(0)]

```

1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v, l)

```

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1
...	[ isEmpty $\mapsto$ False; Value $\mapsto$ 80085; Tail $\mapsto$ ref(0)]

```

1 l = Empty
2 count = int(input("How many elements?"))
3 for i in range(0, count):
4     v = int(input("Insert the next element "))
5     l = Node(v, l)

```

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1	2
...	...	[ isEmpty $\mapsto$ False; Value $\mapsto$ 8078; Tail $\mapsto$ ref(1)]

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Examples of list usage

- We now wish to use the list we just built
- Specifically, we will print all its elements
- **How many elements does it have?**

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Examples of list usage

- We now wish to use the list we just built
- Specifically, we will print all its elements
- **How many elements does it have?**
- Unknown: it is specified by the user!

# Examples of list usage

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

S

PC	I
1	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = 1
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail
    
```

# Examples of list usage

## Lists

## TEAM INFDEV

## Introduction

## Problem discussion

## General idea

## Technical details

## In-class homework

## Conclusion

S

PC	I
1	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```
1 x = 1
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail
```

S

PC	I	x
2	ref(2)	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

What gets printed?

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

What gets printed?  $H[x][Value] = H[2][Value] = 3$



# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

What gets printed?  $H[x][Value] = H[2][Value] = 3$

S

PC	l	x
3	ref(2)	ref(2)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(2)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

**Where is x.Tail?**

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(2)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

Where is **x.Tail**?  $H[x][\text{Tail}] = H[2][\text{Tail}] = \text{ref}(1)$

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(2)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = 1
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

Where is **x.Tail**?  $H[x][\text{Tail}] = H[2][\text{Tail}] = \text{ref}(1)$

S

PC	I	x
4	ref(2)	ref(1)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(1)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

**Where is x.Tail?**

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(1)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

Where is **x.Tail**?  $H[x][\text{Tail}] = H[1][\text{Tail}] = \text{ref}(0)$

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	I	x
3	ref(2)	ref(1)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

```

1 x = 1
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail
    
```

Where is **x.Tail**?  $H[x][\text{Tail}] = H[1][\text{Tail}] = \text{ref}(0)$

S

PC	I	x
4	ref(2)	ref(0)

H

0	1	2
$[E \mapsto T]$	$[E \mapsto F; V \mapsto 2; T \mapsto \text{ref}(0)]$	$[E \mapsto F; V \mapsto 3; T \mapsto \text{ref}(1)]$

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

**What is the value of `x.IsEmpty`?**



# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

**What is the value of `x.IsEmpty`?**  $H[x][IsEmpty] =$   
 $H[0][IsEmpty] = \text{True}$

# Examples of list usage

## Lists

### TEAM INFDEV

#### Introduction

#### Problem discussion

#### General idea

#### Technical details

#### In-class homework

#### Conclusion

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

```

1 x = l
2 while not(x.IsEmpty):
3     print(x.Value)
4     x = x.Tail

```

**What is the value of `x.IsEmpty`?**  $H[x] [IsEmpty] =$   
 $H[0] [IsEmpty] = \text{True}$

S

PC	l	x
5	ref(2)	ref(0)

H

0	1	2
[ E $\mapsto$ T ]	[ E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0) ]	[ E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1) ]

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# In-class homework

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Implement the following (on paper)

- Read a list from the user input
- Remove all odd numbers
- A “volunteer” runs the steps on paper with the memory model

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Implement the following (on paper)

- Read a list from the user input
- Sum all its values
- A “volunteer” runs the steps on paper with the memory model

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Implement the following (on paper)

- Read a list from the user input
- Reverse it
- A “volunteer” runs the steps on paper with the memory model

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Implement the following (on paper)

- Read two lists from the user input
- Append the second to the first (concatenate them)
- A “volunteer” runs the steps on paper with the memory model

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

# Conclusion



Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

## Lecture topics

- What we solved today was the issue of representing multiple data inside a single variable
- We used a simple data structure, the **list**
- We showed how we can consume (use) the list through looping

Lists

TEAM  
INFDEV

Introduction

Problem  
discussion

General idea

Technical  
details

In-class  
homework

Conclusion

The best of luck, and thanks for the  
attention!