# COS 126 Exam Review

- Exams overview
- Example programming exam
- Example written exam questions (part 1)

# Exams tab on the booksite

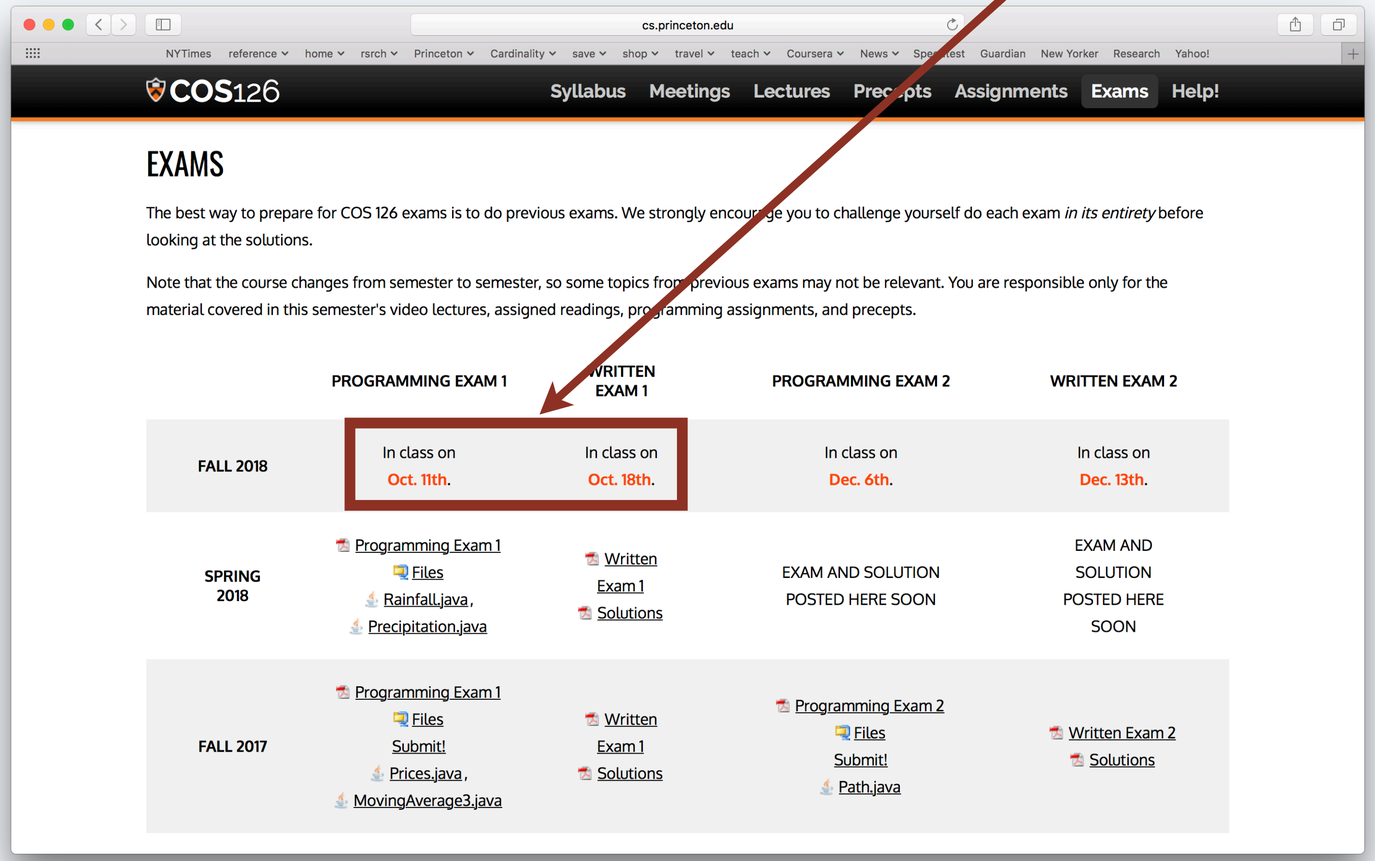
**See Exams tab for full details and old exams.**

- Read carefully *before* each exam.
- Policies are the contract between us and you.

Policies (written exam).

- Closed book/notes/computer.
- 1 page (one side) cheatsheet.
- [two sides for Exam 2.]

***Watch this space for details***



# Things to remember about inclass exams

## We know that you don't have much time.

- Exams are 50 minutes.
- "One page" programming exams.
- Five-minute questions on written exams.

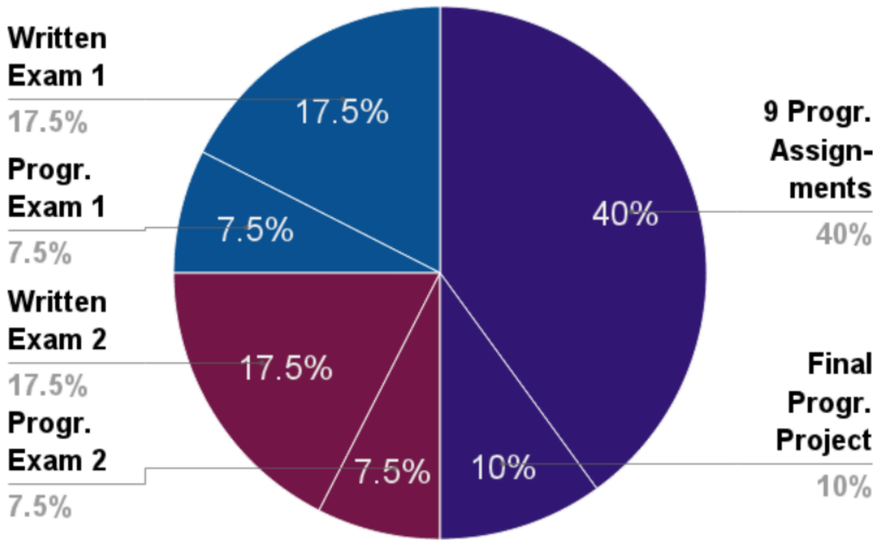
## We have to grade the exams.

- 400+ exams.
- No open-ended questions.
- Fully prepared rubrics.

## Old exams are not completely reliable.

- Course offerings differ slightly.
- We have made mistakes in the past.

## Exams are only part of the story.



# Written Exam Logistics

**The first exam is on Thursday Oct. 18.**

**You don't all fit in this room.**

- Pay attention and know where to go.
- Arrive early.
- No calculator/phone/computer/headphones

**Advice.**

- Review lectures/reading.
- Try an old exam (untimed).
- Try another one (timed).
- Review a few more.

# Example question: Input and output

Q. *Do you understand basic ways of communicating with your programs ?*

**Ex.** ( S2011 Q4) Give the results of invoking this program with the given commands.

```
public class Q4
{
    public static void main(String[] args)
    {
        int curr = StdIn.readInt();
        StdOut.print(curr + " ");
        int prev = curr;
        while (!StdIn.isEmpty())
        {
            curr = StdIn.readInt();
            StdOut.print((prev + curr) / 2 + " ");
            prev = curr;
        }
        StdOut.println();
    }
}
```

```
% more input.txt
2 4 6 8 10 12 8 2
% java Q4 < input.txt
```

```
2 3 5 7 9 11 10 5
```

```
% java Q4 < input.txt | java Q4
```

```
2 2 4 6 8 10 10 7
```

*Note: It prints the first number, then the average of each number and its predecessor.*

# Example question: Functions

Q. *Do you understand basic mechanisms for invoking functions ?*

**Ex.** ( S2018 Q7) Give the contents of the array a[] after executing the given code.

```
public static int halve1(int x)
{
    x = x / 2;
    return x;
}

public static void halve2(int[] a)
{
   for (int i = 0; i < a.length; i++)
    {
        halve1(a[i]);
        a[i] = halve1(a[i]);
    }
}
```

```
int[] a = { 16, 32, 48, 64 };
halve2(a);
```

8 16 24 32 NOT 4 8 12 16

```
int[] a = { 16, 32, 48, 64 };
halve2(a);
halve2(a);
```

4 8 12 16

# Example question: Functions

**Ex.** ( S2018 Q7) Give the contents of the array a[] after executing the given code.

```
public static void halve3(int[] a)
{
    int n = a.length;
    int[] b = new int[n/2];
    for (int i = 0; i < n/2; i++)
        b[i] = a[i];
    a = b;
}
```

```
int[] a = { 16, 32, 48, 64 };
halve3(a);
halve3(a);
```

**16 32 48 64**

# Example question: Recursion

Q. *Can you figure out the effect of a simple recursive program (or two)?*

**Ex.** ( Fall 2017 Q5) Fill in the values returned by these mutually recursive functions:

```
public static int mystery1(int n)
{
   if (n == 0) return 0;
   else return mystery2(n - 1);
}
public static int mystery2(int n)
{
   if (n == 0) return 1;
   else return mystery1(n - 1);
}
```

| n | mystery1(n) | mystery2(n) |
|---|---|---|
| 0 | **0** | **1** |
| 1 | **1** | **0** |
| 2 | **0** | **1** |
| 3 | **1** | **0** |
| 4 | **0** | **1** |
| 5 | **1** | **0** |

Write *one line of code* that could replace the body of `mystery(1)`.

```
return n % 2;
```

# Example question: Binary operations

Q. Why is **~0** equal to **-1** and not **1**? (Fall 2014 Q1B)

A (wrong).

~ is "not"

0 is "false"

"not false" is “true"

"true" is 1

A (correct).

~ is BITWISE "not"

0 is 00000000000000000000000000000000

~0 is 11111111111111111111111111111111

11111111111111111111111111111111 is -1 (2s complement)

# Example question: TOY/number representation

Q. (Fall 2013 Q8) Consider this sequence of TOY instructions:

| | | |
|---|---|---|
| **7 1 0 1** | `R[1] = 0001` | |
| **2 2 0 1** | `R[2] = R[0] - R[1]` | sets R[2] to all 1s |
| **4 7 7 2** | `R[7] = R[7] ^ R[2]` | sets R[7] to bitwise XOR of R[7] with all 1s |
| **1 7 7 1** | `R[7] = R[7] + R[1]` | adds 1 to R[7] |

Q. What is the value of R[7] after this sequence if it was intially 0025 ?

**0000000000100101**
**1111111111011010**
**1111111111011011**

**FFDB**

Q. In English, what does sequence do to R[7] ?

**Negates it.**

to negate a 2s complement number: *flip its bits and add 1*

# Example question: TOY

Q. *Can you simulate the effect of a very simple TOY program*?

**Ex.** ( Fall 2016 Q7) Suppose that you load the following into memory locations 10-17 of TOY, set the PC to 10, and press RUN. Give the result in 01 when 00 is initially **0001**.

```
10: 8A00 R[A] <- M[00]
11: 7101 R[1] <- 1
12: 221A R[2] <- R[1] - R[A]
13: D216 if (R[2] > 0) PC <- 16
14: 1111 R[1] <- R[1] + R[1]
15: C012 PC <- 12
16: 9101 M[01] <- R[1]
17: 0000 halt
```

| PC | R[A] | R[1] | R[2] | |
|---|---|---|---|---|
| 10 | 0001 | | | |
| 11 | 0001 | 0001 | | |
| 12 | 0001 | 0001 | 0000 | |
| 13 | 0001 | 0001 | 0000 | |
| 14 | 0001 | 0002 | 0000 | |
| 12 | 0001 | 0002 | 0001 | >0 |
| 13 | 0001 | 0002 | 0001 | |
| 16 | 0001 | 0002 | 0001 | |
| 17 | 0001 | **0002** | 0001 | |

# Example question: TOY

Q. *Can you simulate the effect of a simple TOY program*?

**Ex.** ( Fall 2016 Q7) Suppose that you load the following into memory locations 10-17 of TOY, set the PC to 10, and press RUN. Give the result in 01 when 00 is initially **0006**.

```
10: 8A00 R[A] <- M[00]
11: 7101 R[1] <- 1
12: 221A R[2] <- R[1] - R[A]
13: D216 if (R[2] > 0) PC <- 16
14: 1111 R[1] <- R[1] + R[1]
15: C012 PC <- 12
16: 9101 M[01] <- R[1]
17: 0000 halt
```

| PC | R[1] | R[2] |
|---|---|---|
| 10 | | |
| 11 | 0001 | |
| 12 | 0001 | FFFA |
| 13 | 0001 | FFFA |
| 14 | 0002 | FFFA |
| 12 | 0002 | FFFE |
| 13 | 0002 | FFFE |
| 14 | 0004 | FFFE |

*0001–0006*

*<0*

| PC | R[1] | R[2] |
|---|---|---|
| 12 | 0004 | FFFC |
| 13 | 0004 | FFFC |
| 14 | 0008 | FFFC |
| 12 | 0008 | FFF9 |
| 13 | 0008 | |
| 14 | 0010 | FFF9 |
| 12 | 0010 | 0004 |
| 13 | 0010 | 0004 |
| 16 | **0010** | 0004 |

*NOT 0016*

*>0*

# Example question: TOY

Q. *Can you **reason about** the effect of a simple TOY program*?

**Ex.** ( Fall 2016 Q7) Suppose that you load the following into memory locations 10-17 of TOY, set the PC to 10, and press RUN. Give the result in M[01] when M[00] is initially **1EAF**.

```
10: 8A00 R[A] <- M[00]
11: 7101 R[1] <- 1
12: 221A R[2] <- R[1] - R[A]
13: D216 if (R[2] > 0) PC <- 16
14: 1111 R[1] <- R[1] + R[1]
15: C012 PC <- 12
16: 9101 M[01] <- R[1]
17: 0000 halt
```

```
load limit from M[00]
x = 1
while (x <= limit)
{
    x = 2*x
}
store x to M[01]
```

1
2
4
8
10
20
40
80
100
...
**2000**

# Good luck!