Review sessions, week of 7 May, all in Phillips 101

Day	Time	Instructor room	Topic
Mon	1PM	Liviu Popescu	Executing sequences of statements that involve creating new objects
Mon	2PM	Bruno Abrahao	Writing constructors in classes and subclasses
Mon	3PM	Sharmin Azam	Casting, apparent and real classes
Tue	1PM	David Gries	Developing loops from invariants
Tue	2PM	David Gries	Developing required algorithms
Tue	3PM	Gurmeet Singh	Matlab
Wed	1PM	Asif Sheikh	Specific classes: e.g. Integer, Character, Vector, String
Wed	2PM	David Gries	Executing method calls; frame for a call
Wed	3PM	Piti Irawan	Recursion

The final is cumulative, covering all topics in the course except as described below. So, you have to know everything that was covered in the three prelims. See the handouts on the three prelims (on the course web page).

You do not have to study the following topics: Exception handling, abstract classes, reading a file or the keyboard, applications, applets.

You have to know:

1. **Multi-dimensional arrays**. See below.

2. **Placement of components in a GUI**. The default layout managers for a `JFrame`, a `JPanel`, and a `Box` and how that manager arranges components in it. What these basic components are: `JButton`, `JLabel`, `JTextField`, `JTextArea`. You do not have to know how to "listen" to an event.

3. **Matlab.** See below.

4. **Several algorithms**. You know this already, but we repeat it for emphasis. one of the following algorithms can be asked for. We may simply write "show binary search", or "Show us the partition algorithm", and you have to give the precondition, postcondition and loop invariant and then develop the algorithm. Expected: the loop with initialization is developed from the invariant; a loop that has nothing to do with the invariant you write gets little credit. Everyone should get full credit on this question because it is simply a matter of practicing developing known algorithms from their specs.

Linear search, Binary search, Dutch National Flag, Partition algorithm, Selection sort, Insertion sort.

5. **Multi-dimensional arrays**. You have to know about rectangular arrays and ragged arrays (in which rows may have different lengths). This includes knowing how to access the number of columns in a given row and knowing how to create a rectangular array and a ragged array. You have to know how arrays are stored as objects (folders) and to be able to draw an array, as done on prelim 2.

MATLAB

0. The notation `a:b` to denote the row of integers `[a (a+1) (a+2) ... b]`. The notation

```
a : increment : b
```

1. The basics of creating arrays, using

(a) row vector notation: `[ 10  20  30 ]`

(b) column vector notation `[10; 20; 30]`

(c) rectangular array notation

```
[10 20 30; 40 50 60]
```

(d) stacking rows: if x is `[10 20]`,

then `[x x]` is `[10 20 10 20]`

(e) stacking columns. if `u = [1 2]; v = [3 4]` then `[u; v; u]` is

```
1 2
3 4
1 2
```

(f) for a row vector, `size(x)` is the number of elements. For an array, `size(x)` is the number of elements in each dimension. For this purpose, a column vector is really a 1-by-n array.

(g) transpose `b'` of an array or vector `b`

(h) functions `zeros(n)`, `zeros(n,m)`, `ones(n)`, and `ones(n,m)`

(i) elementwise addition, subtraction, multiplication, division of an array by a scalar or a scalar by an array, e.g. `[10 5 2] + 5`.

(j) Elementwise operations on arrays of the same shape and size:

addition `b + c`
subtraction `b – c`
multiplication `b .* c`
division `b ./ c`
exponentiation `b .^ c`

(k) functions `max`, `min`, `sum`, `cumsum`, `prod`, `cumprod`, `median`, `abs`, `floor`, `ceil`, `sqrt`

(l) Defining functions, e.g.

```
% = mean & standard dev. of nonempty row vector x
function [mean,stdev] = stat(x)
  n= length(x);
  mean= sum(x)/n;
  stdev= sqrt(sum((x-mean).^2)/n);
```

(m) if-statements, e.g.

```
 d= sqrt(b^2 -4*a*c);
  if d>0
      r1= (-b+d)/(2*a);
      r2= (-b-d)/(2*a);
  else
      disp(`Complex roots')
  end

  if (x>y) & (x>z)
      maxval= x;
  elseif y>z
      maxval= y;
  else
      maxval= z;
  end
```

(n) Be able to put together expressions that calculate a sum (or product) or cumulative sum (or product) of n terms of a series. Typically, there are constructed from (1) base arrays using functions zeros, ones, rand, 1:n, and linspace and (2) elementwise array operations.

For example, we wrote this code for the cumulative sum of n approximations to pi using Wallis's formula

```
pi/2=  2*2/(1*3) + 4*4/(3*5) + ...
```

```
  evens= 2 * (1:n)
  odds= evens – 1
  answer= 2*cumsum((evens .^ 2) ./
          (odds .* (odds + 2))
```

Here are some infinite sums to practice on.

```
5 + 5 + 5 + 5 + ...
```

```
1 - 1 + 1 - 1 + 1 - 1 + ...
```

```
1/(1*2*3) + 1/(3*4*5) +
                    1/(4*5*6) + ...
```

```
1/1 + 1/2 + 1/3 + 1/4 + ...
```

```
1/(1*1) + 1/(2*2) + 1/(3*3) +
                    1/(4*4) + ...
```

```
1/1 - 1/2 + 1/3 - 1/4 +
                    1/5 - 1/6 + ...
```

```
1*2/(2*3) + 2*3/(3*4) +
                    3*4/(4*5) + ...
```