Faculty of Science
COMP-202B - Foundations of Computing (Winter 2016) - All Sections
Midterm Examination

February 23rd, 2016
18:00 - 21:00

Examiners: Yang Cai [Section 1 TR (10:00-11:30)]
Jackie Chi Kit Cheung [Section 2 MW (10:00-11:30)]
Melanie Lyman-Abramovitch [Section 3 MWF (12:30-13:30)]

Instructions:

- **DO NOT TURN THIS PAGE UNTIL INSTRUCTED**
- This is a **closed book** examination; only a letter-sized (8.5" by 11") **crib sheet** is permitted. This crib sheet can be single or double-sided; it can be handwritten or typed. Non-electronic translation dictionaries are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- Besides the above, only writing implements (pens, pencils, erasers, pencil sharpeners, etc.) are allowed. The possession of any other tools or devices is prohibited.
- Answer **all** true/false and multiple choice questions on the scantron sheet.
- Answer questions 41 and 42 in the booklet provided.
- You may take your question sheet home. We recommend marking your answers down so that you may verify them against the posted solutions.
- This examination has **17** pages including this cover page, and is printed on both sides of the paper. On page 16, you will find information about **useful classes and methods**. **You may detach the Appendix from the examination if you wish.**
- The Examination Security Monitor Program detects pairs of students with unusually similar answer patterns on mulitple-choice exams. Data generated by this program can be used as admissible evidence, either to initiate or corroborate an investigation or a charge of cheating under Section 16 of the Code of Student Conduct and Disciplinary Procedures.
- **MAKE SURE TO WRITE YOUR NAME AND STUDENT ID ON THE SCANTRON AS WELL AS THE BOOKLET FOR THE LONG ANSWER PROBLEMS. PLEASE WRITE LEGIBLY**

Scoring

The exam will be scored as follows:

1. Questions 1 to 10 are worth 1 point each
2. Questions 11 to 40 will be worth 2 points each
3. Questions 41 and 42 are worth 15 points each

Version-1

True/False Section (1 point each)

1. In Java, the value of the expression `7/4` is 1.

   (A) TRUE

   (B) FALSE

2. In Java, a `void` method can return a value using a print statement.

   (A) TRUE

   (B) FALSE

3. In Java, a method can have more than one input argument.

   (A) TRUE

   (B) FALSE

4. Any integer CAN be represented using an `int` type variable.

   (A) TRUE

   (B) FALSE

5. What is the value of the boolean variable `b`?

```
boolean b = true && (!false || true);
```

   (A) `true`

   (B) `false`

6. You can put more than one if block within the same for loop.

   (A) TRUE

   (B) FALSE

7. You can declare more than one variable with the same name within the same method.

   (A) TRUE

   (B) FALSE

8. Bytecode is the result of compiling a Java program into a form that can be interpreted by the Java Virtual Machine.

   (A) TRUE

   (B) FALSE

9. Variable assignment is an example of a control flow statement that affects which lines of code in a program are executed.

   (A) TRUE

   (B) FALSE

10. It is possible to represent complex data such as videos using only binary numbers.

    (A) TRUE

    (B) FALSE

Regular Multiple Choice (2 points each)

11. What are the values of x, y, and z after the following code executes?

```
int x = 5;
int y = 6;
int z = 7;
if(x<y){
   x=x+1;
}
else if(x==y){
   y=y+1;
}
else if(y==z){
   z=z+1;
}
else{
   z=x;
}
```

(A) x=6, y=6, z=7

(B) x=6, y=6, z=6

(C) x=5, y=6, z=5

(D) x=6, y=7, z=8

(E) x=5, y=6, z=5

12. What are the values of x,y, and z after the following code executes?

```
int x = 5;
int y = 6;
int z = 7;
x = y;
y = y + 5;
z = z / 2;
z++;
--y;
```

(A) x=6, y=10, z=4

(B) x=10, y=10, z=5

(C) x=10, y=9, z=5

(D) x=6, y=10, z=5

(E) x=5, y=6, z=7

13. What does the following snippet of code print?

```
String s = "apple";
for(int i=0; i<s.length(); i++){
   System.out.print((char) s.charAt(i+1));
}
```

(A) pple

(B) This is an infinite loop - it prints forever

(C) There is a run-time error

(D) apple

(E) There is a compile-time error

14. Consider the following method:

```
public static void method(String s){
  String t = s+s;
  System.out.println(t);
}
```

Which of the following is a correct way to call this method?

(A) All of the options are valid.

(B) `System.out.println(method("Apple Juice"));`

(C) `method("Apple Juice");`

(D) `String t = method("Apple Juice");`

(E)
```
String s = "Kittens";
method();
```

15. What does the following method do?

```
public static int method(String s){
   int x = 0;
   for(int i=0; i<s.length(); i++){
     if(s.charAt(i)>='0' && s.charAt(i) <='9'){
       x+= (s.charAt(i) - '0');
     }
   }
   return x;
}
```

(A) returns the number of non-zero digits in a String.

(B) returns the sum of all numbers in a String

(C) returns the sum of the ASCII values of all the numbers in a String

(D) returns the number of digits in a String

(E) Shifts all characters in a String backwards by a value of '0'

16. How many times is the condition of the following for loop checked?

```
for (int i = 0; i < 5; i += 3) {
  System.out.println("Repeat!");
}
```

(A) 3

(B) 0

(C) 2

(D) 4

(E) 1

17. Suppose you have a method with the following header.

```
public static int fun(double x, int y)
```

Which of the following are legal ways to call the method?

```
I.   fun(5, 6.0);
II.  fun(5, "6");
III. fun(5, 6);
```

(A) I only

(B) None of them

(C) III only

(D) I and III

(E) II only

18. How many times will the following code print ”Hi!”?

```
for (int i = 0; i < 6; i++){
  for (i = 0; i < 6; i++){
    System.out.println("Hi!");
  }
}
```

(A) 36

(B) 25

(C) 6

(D) 0

(E) 5

19. What will be printed when the following code is run?

```
public static void main(String[] args) {
  int[] arr1 = {2, 3};
  int[] arr2 = {1, 4};
  foo(arr1, arr2);
  System.out.print(arr1[0] + " " + arr1[1] + " " + arr2[0] + " " +
      arr2[1]);
}

public static void foo(int[] a, int[] b) {
  a[0] = b[1];
  a[1] = b[0];
  b[0] = a[1];
  b[1] = a[0];
}
```

(A) 4 1 1 4

(B) 2 3 1 4

(C) 1 4 2 3

(D) 3 2 2 3

(E) 4 1 3 2

20. What will be printed when the following code is run?

```
public static void main(String[] args) {
  int x = 5;
  int y = 10;
  int z = 15;
  swap(x, y);
  swap(y, z);
  swap(z, x);
  System.out.print(x + " " + y + " " + z);
}

public static void swap(int x, int y) {
  int temp = x;
  x = y;
  y = temp;
}
```

(A) 15 5 10

(B) 5 10 15

(C) 5 15 10

(D) 10 5 15

(E) 10 15 5

21. What does the following expression evaluate to (data type and value)?

```
1 + 2 + ""
```

(A) int, 12

(B) This code will cause a run-time error.

(C) String, "12"

(D) int, 3

(E) String, "3"

22. What is the result of executing the following piece of code?

```
int [] arr = {1, 2, 3, 4, 5};
for (int i = 0; i < arr.length; i++) {
  arr[i] = i + 1;
}
```

(A) It will do nothing because the body of the for loop is never executed.

(B) It will cause a compile-time error.

(C) Each element of `arr` will have its value increased by 1.

(D) The elements of `arr` will remain the same.

(E) It will cause a run-time error.

23. I would like to create a new data type, `Complex`, which stores *complex numbers*. The complex numbers represented by my data type take the form $a + bi$, where a and b are `double` values, and i is a constant [a]. Given that a `double` value in Java requires 64 bits of memory, how many bits are needed to represent a `Complex` number?

(A) 128

(B) 64

(C) 193

(D) 192

(E) 129

[a] For those familiar with complex numbers, i is equal to the square root of -1. For those unfamiliar, yes, this is real thing in physics, math and engineering!

24. What does the following code fragment do?

```
int total = 0;
for (int x = 10; x > 0; x = x - 2){
   total = total + x;
}
```

(A) Nothing. There is a compile time error.

(B) Calculates the difference between adjacent even numbers from 1 to 10 (inclusive).

(C) Calculates the sum of all the even numbers strictly less than 10.

(D) Calculates the sum of all the positive even numbers less or equal to 10.

(E) Calculates the sum of all the numbers between 1 and 10 (inclusive).

25. What prints of the code fragment given below?

```
int i;
int j = 0;
for (i = 0; i < 4; i++){
  if (i % 2 == 0){
    i = i + 2;
    j++;
  }
  else{
    i++;
    j = j + 2;
  }
  j++;
}
System.out.println("i=" + i + ", j=" + j);
```

(A) i=4, j=5

(B) i=6, j=5

(C) i=4, j=6

(D) i=6, j=6

(E) i=5, j=5

26. An array is defined as

```
int [][] arr = { {3, 4, 5}, {1, 2, 9, 7}, {6, 7} };
```

What is the value of arr.length?

(A) 4

(B) 12

(C) It is undefined.

(D) 9

(E) 3

27. An array is defined as

```
int [][] arr = { {3, 4, 5}, {1, 2, 9, 7}, {6, 7} };
```

What is the value of arr[1].length?

(A) 9

(B) 12

(C) 4

(D) 3

(E) It is undefined.

28. What numbers does the following code output?

```
int [] numbers = {1, 2, 3, 4, 5, 6, 7, 8};
for (int i = 0; i < 7; i ++){
   System.out.print(numbers[7-i] + " ");
}
```

(A) 8 7 6 5 4 3 2

(B) 7 6 5 4 3 2 1

(C) 0 1 2 3 4 5 6 7

(D) 1 2 3 4 5 6 7 8

(E) There is a run-time error.

29. What is printed by the following code segment?

```
int[] numbers = new int[8];
numbers[0] = 0;
for (int i = 0; i < (numbers.length-1); i++){
   numbers[i+1]=numbers[i]+i;
}
System.out.println(numbers[7]);
```

(A) 8

(B) 15

(C) 13

(D) 7

(E) 21

30. What is printed by the segment of code below?

```
public static void main (String[] args){
  int x = 3;
  int y = f(x);
  System.out.print(y);
}
public static int f(int y){
  System.out.print("y=");
  return y + 3;
}
```

(A) There is a compile-time error.

(B) y=x+3

(C) y=6

(D) x=6

(E) x=y

31. What is/are the value(s) returned in the following method?

```
public static int getNum(){
  int x = 3;
  if (x < 5){
    return 5;
  }
  return x;
}
```

(A) 5 and 3

(B) There is a compile-time error.

(C) 8

(D) 5

(E) 3

32. What will the following code print?

```
int [][] arr = {{1,2,3},{4,5,6},{7,8}};
for(int i=0; i<arr.length; i++){
  for(int j=0; j<arr[0].length; j++){
    System.out.print(arr[j][i] + " ");
  }
}
```

(A) There is a run-time error

(B) 1 2 3 4 5 6 7 8

(C) 7 8 4 5 6 1 2 3

(D) 1 4 7 2 5 8 3 6

(E) 8 7 6 5 4 3 2 1

33. What is the decimal number 17 in binary?

    (A) 1001

    (B) 10010

    (C) 100001

    (D) 10001

    (E) 1111

34. What is the maximum (decimal) value that can be represented by a five-digit binary number?

    (A) 31

    (B) 5

    (C) 16

    (D) 64

    (E) 32

35. What is the minimum number of bits required to represent the decimal number 444?

    (A) 9

    (B) 444

    (C) 8

    (D) 2

    (E) 222

Multiple Answer Multiple Choice (2 points each)

In this section, the questions may have more than one correct answer. In order to get full marks, shade in *all* correct answers on your scantron sheet. No part marks will be given.

For the next two questions, consider the following method (note, this method may contain errors):

```
public static int method(int x){
   ****Starred line 1*****
   for(int i=0; i<Math.abs(x); i++){
      int y = x/2;
      ****Starred line 2****
      if(x<y){
         System.out.print(x*y);
         ****Starred line 3****
      }
      ****Starred line 4****
   }
   ****Starred line 5****
}
```

36. The variable y exists at which of the starred lines?

    (A) 1

    (B) 2

    (C) 3

    (D) 4

    (E) 5

37. The variable i exists at which of the starred lines?

    (A) 1

    (B) 2

    (C) 3

    (D) 4

    (E) 5

38. Which of the following is a boolean expression in Java?

    (A) a && b

    (B) a <> b

    (C) a = b

    (D) a := b

    (E) a < 5 && > 0

39. Let `a` and `b` be boolean variables. Which of the following expressions is equivalent to `a || b`? (Two logical expressions are equivalent if whenever one of them is true, so is the other, and vice versa.)

    (A) `(a && b) || (a && !b) || (!a && b)`

    (B) `(a && b) || (a && !b) || (!a && !b)`

    (C) `(a && !b) || (!a && b) || (!a && !b)`

    (D) `(a && !b) || (!a && b)`

    (E) `!(a && b)`

40. Select all of the following which are examples of algorithms.

    (A) The weather forecast for the next week.

    (B) A recipe for baking a cake.

    (C) The COMP-202 course outline.

    (D) An IKEA catalogue with pictures and prices of furniture for sale.

    (E) A series of steps on a menu for ordering a custom combo at a restaurant.

Coding Section

Answer the following questions by writing your solutions in the provided booklet.

41. **Product of Two Prime Numbers**

Write a method called `productOfTwoPrimes`, which takes a positive integer as input and returns `true` if the input is the product of exactly two prime numbers and otherwise returns `false`. You may assume you have access to a method whose signature is `public static boolean isPrime(int x)` that determines whether or not the input value, x, is a prime number. Recall that a number is prime if it is only evenly divisible by itself and 1. For this question, the numbers 0 and 1 are *not* considered to be prime.

For example, `productOfTwoPrimes(9)` should return `true` because 9 is the product of 3 and 3. On the other hand, `productOfTwoPrimes(63)` should return `false` because it is the product of three primes (3, 3, and 7).

42. **Second Largest int**

For this problem, you may not use any methods from the Java standard libraries (e.g., methods from the Math or Arrays libraries).

Write a method called `secondLargest`, which takes an int array as input, and returns the second-largest value in the int array. You may make the following assumptions:

- The input array has at least two elements.
- All elements of the input array are unique (i.e., there are no duplicates).

For example, calling the method on an array containing the values `{3, 7, 5, 8, 2}` should return 7.

SUMMARY OF JAVA STANDARD LIBRARY METHODS FOR SELECTED CLASSES

- `Arrays` (package `java.util.Arrays` Methods:
  - `public int[] copyOfRange(int[] original, int from, int to)`: Returns a subset of the `original` starting at `from` and finishing at `to`, excluding `to`. `to` might lie outside of the array.
- `String` (package `java.lang`) Methods:
  - `public boolean equals(Object anObject)`: Compares this `String` to `anObject`.
  - `public int length()`: Calculates the length of this `String`.
  - `public char charAt(int i)`: Gets the `char` at position `i` of the `String`. Note that counting starts from 0 so that to get the first character of the `String` you should input `i` equals 0.
  - `public boolean equalsIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString`.
  - `public int compareTo(String anotherString)`: Compares this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int compareToIgnoreCase(String anotherString)`: Compares, ignoring case considerations, this `String` to `anotherString` lexicographically; returns a negative value if this `String` occurs before `anotherString`, a positive value if this `String` occurs after `anotherString`, and `0` if both `String`s are equal.
  - `public int indexOf(String str)`: Returns the index within this string of the first occurrence of the specified substring. If no such substring exists, returns -1.
  - `public int lastIndexOf(String str)`: Returns the index within this string of the last occurrence of the specified substring. If no such substring exists, returns -1.
  - `public String replace(char c, char d)` : Returns a new String with all occurrences of the character `c` in the `this` String replaced by the character `d`.
  - `public String replace(CharSequence target, CharSequence replacement)`: Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. (N.B.: A String is a kind of CharSequence.)
  - `public char[] toCharArray()`: Converts this `String` to a new character array.
  - `public String toLowerCase()`: Converts all of the characters in this String to lower case.
  - `public String[] split(String regex)`: Splits this string around matches of the given regular expression.
  - `public String substring(int start, int finish)`: Returns a new String composed of the `this` String starting from index `start` and up to, but not including index of `finish`
  - `public String toUpperCase()`: Converts all of the characters in this String to upper case.
- `File` (package `java.io`) Methods:
  - `public FileSstring pathname)`: Creates a new `File` instance that corresponds to the given pathname.
- `Scanner` (package `java.util`) Methods:
  - `public Scanner(Inputstream source)`: Constructs a new `Scanner` that produces values scanned from the specified input stream.
  - `public Scanner(File f)`: Constructs a new `Scanner` that produces values scanned from the specified File
  - `public double nextDouble()`: Scans the next token of the input as a `double`.
  - `public boolean nextBoolean()`: Scans the next token of the input as a `boolean`.
  - `public int nextInt()`: Scans the next token of the input as an `int`.
  - `public String nextLine()`: Advances this `Scanner` past the current line and returns the input read.
  - `public boolean hasNextLine()`: Checks whether there are further lines left to scan.
- `PrintStream` (package `java.io`) Methods:
  - `public void print(boolean b)`: Prints `boolean` value `b`.
  - `public void print(double d)`: Prints `double` value `d`.
  - `public void print(int i)`: Prints `int` value `i`.
  - `public void print(Object o)`: Prints `Object` `o`.
  - `public void print(String s)`: Prints `String` `s`.
  - `public void println()`: Terminates the current line by writing the line separator string.
  - `public void println(boolean b)`: Prints `boolean` value `b` and then terminates the line.
  - `public void println(double d)`: Prints `double` value `d` and then terminates the line.
  - `public void println(int i)`: Prints `int` value `i` and then terminates the line.
  - `public void println(Object o)`: Prints `Object` `o` and then terminates the line.
  - `public void println(String s)`: Prints `String` `s` and then terminates the line.
- `Math` (package `java.lang`) Methods:

- `public static double pow(double a, double b)`: Returns the value of `a` raised to the power of `b`.
- `public static double sqrt(double a)`: Returns the correctly rounded positive square root of `double` value `a`.
- `public static double random()`: Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.
- `public static double exp(double a)`: Returns Euler's number e raised to the power of `double` value `a`. (base e) of `double` value `a`. of `double` value `a`.

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

Figure 1: ASCII Table