CS180 Spring 2010 Final Exam 8 May, 2010
Prof. Chris Clifton

Turn Off Your Cell Phone. Use of any electronic device during the test is prohibited.

Time will be tight. If you spend more than the recommended time on any question, **go on to the next one**. If you can't answer it in the recommended time, you are either going in to too much detail or the question is material you don't know well. You can skip one or two parts and still demonstrate what I believe to be an A-level understanding of the material.

Solutions in boldface, *scoring methodology in italics*. My expectations were than an A student should get at least a 43 and a B student at least 34. I would be seriously concerned about someone who didn't get at least a 14 being ready for follow-on work (i.e., the C- cutoff.)

Multiple Choice

Provide answers on the Scantron card. (You may mark them on this sheet as well, but the mark on the Scantron card is the official version.)

1 Constructors and Inheritance (4 minutes, 2 points)

Given the following classes:

```
class A {
  public A() {
    System.out.print("1");
  }
}

class B extends A {
  public B() {
    System.out.print("2");
  }
}

class C extends B {
  public C() {
    System.out.print("3");
  }
}

public class TestOne{
  public static void main(String args[]){
    C c = new C();
  }
}
```

What is the result when TestOne is executed?

A 3

B 1

C 321

D **123**

2 Inheritance and overriding (5 minutes, 2 points)

Given the following code:

```
public class SimpleCalc {
  public int value = 0;
  public void calculate() { value += 7; }
}

public class MultiCalc extends SimpleCalc {
  public int value = 0;
```

```
  public void calculate(){ value = value - 3; }

  public void calculate(int multiplier) {
    calculate();
    super.calculate();
    value = value * multiplier;
  }

  public static void main(String args[]) {
    MultiCalc calculator = new MultiCalc();
    calculator.calculate(2);
    System.out.println("Value is: " + calculator.value);
  }
}
```

What is the result when the above is run (that is, the main method is called)?

A Compilation fails

B **Value is: -6**

C Value is: 12

D Value is: -12

E The code runs with no output

3 Threads (4 minutes, 3 points)

```
public class TestOne implements Runnable {
  public static void main(String args[]) throws Exception {
    Thread t = new Thread(new TestOne());
    t.start();
    System.out.print("Started");
    t.join();
    System.out.print("Complete");
  }

  public void run() {
    for(int i = 0; i < 4; i++){
      System.out.print(i);
    }
  }
}
```

Given the above code, which of the following is a possible result?

A Compilation fails

B An exception is thrown at runtime

C The code executes and prints "StartedComplete"

D The code executes and prints "StartedComplete0123"

E **The code executes and prints "Started0123Complete"**

4 Overloading / signatures (4 minutes, 1 point)

Which of the following would overload this method:

```
double add(int a, int b) { return a + b; }
```

```
I.   int add(int a, int b) { return a + b; }
II.  double add(double a, double b) { return a + b; }
III. int add(double a, double b) { return (int)(a + b); }
IV.  double add(int a, int b) { return a + b; } in a child class.
V.   int add(int a, int b) { return a + b; } in a child class.
```

A I, II

B I, III

C **II, III**

D IV, V

E IV only

5 Arrays (6 minutes, 2 points)

What is the output of the following program:

```
public class MyProgram2 {
  public static int id;
  public MyProgram2(int d) {
    id = d;
  }

  public int getID() {
    return id;
  }

  public static void main(String[] args) {
    MyProgram2[] prog = new MyProgram2[5];
    for(int i = 0; i < prog.length; i++) prog[i] = new MyProgram2(i);
    for(int i = 0; i < prog.length; i++) System.out.print(prog[i].getID());
  }
}
```

A 01234

B 55555

C **44444**

D 43210

E 00000

6 Arrays vs. Linked Lists (3 minutes, 2 points)

Which of the following operations would be faster with an array than with a linked-list data structure:

A Adding an element in between existing elements.

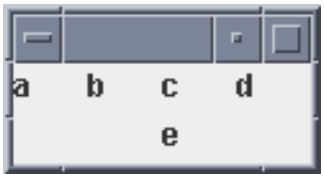
B Changing the value of the first element.

C **Getting the value of an element in the middle.**

D Removing an element from the middle.

E None of these would be faster using arrays.

7 Layout Managers (5 minutes, 1 point)



Which of the following will create the Window shown above?

A
```
setLayout(new FlowLayout());
add(new JLabel("a"));
add(new JLabel("b"));
JPanel j = new JPanel();
j.setLayout(new GridLayout(2,2));
j.add(new JLabel("c"));
j.add(new JLabel("d"));
j.add(new JLabel("e"));
add(j);
```

B **The following is the correct answer. This was pretty easy for people who put a lot of extra work in Project 4 (for which they didn't get much credit.)**

```
setLayout(new GridLayout(2,4));
add(new JLabel("a"));
add(new JLabel("b"));
add(new JLabel("c"));
add(new JLabel("d"));
add(new JLabel(""));
add(new JLabel(""));
add(new JLabel("e"));
```

C
```
setLayout(new BorderLayout());
JPanel i = new JPanel();
i.add(new JLabel("a"));
i.add(new JLabel("b"));
JPanel j = new JPanel();
j.setLayout(new GridLayout(2,2));
j.add(new JLabel("c"));
```

```
        j.add(new JLabel("d"));
        j.add(new JLabel("e"));
        i.add(j);
        add(i, BorderLayout.CENTER);
```

D Both A and C

E A, B, and C would all create the window as shown.

8 Exceptions (3 minutes, 2 points)

Assume we have defined the following exceptions:

```
public class AException extends Exception {}
public class BException extends AException {}
public class CException extends BException {}
```

and we use them in the following program:

```
public static void main(String[] args)
{
   try {
      FuncA(true);
      System.out.print("1");
   }
   catch(AException ex) {
      System.out.print("2");
   }
   catch(Exception ex) {
      System.out.print("3");
   }
   System.out.print("4");
}
```

```
static void FuncA(boolean x) throws Exception
{
   try {
      if(x) throw new BException();
   }
   catch(CException ex) {
      System.out.print("5");
      throw new AException();
   }
   System.out.print("6");
}
```

What should the output of the program be?

A **24**

B 134

C 524

D 5124

E None of the above.

9 Using exceptions (3 minutes, 1 point)

The exceptions in question 8 give no information other than the class of the exception. However, it is possible to put more information in an exception. Which of the following would NOT be an appropriate use of an exception carrying extra information:

A **To write a function that always returns two values (since "return" only lets us return one.)**

B To give a message about what caused an error.

C When in an unusual situation, we want to return an item of a different type from the normal return.

D When the object to be returned has a problem that prevents it from being used normally, but we still want to give it to the calling method.

10 Recursion (8 minutes, 3 points)

What does method recur do, when called as recur(x, x.length) ?

```
//Precondition: x is an array of n integers
public static int recur (int[] x, int n) {
  if (n == 1) return x[0];
  else {
    int t = recur(x, n - 1);
    if(x[n-1] > t) return x[n-1];
    else return t;
  }
}
```

A **It finds the largest value in x and leaves x unchanged.**

B It finds the smallest value in x and leaves x unchanged.

C It sorts x in ascending order and returns the largest value in x.

D It sorts x in descending order and returns the largest value in x.

E It returns x[0] or x[n-1], whichever is larger.

11 Recursion basics (2 minutes, 2 points)

A recursive program must have:

A An Array to operate on.

B **At least one base case and at least one recursive case.**

C A Linked-List data structure to operate on.

D A counter to decide how many times to recurse.

E At least one for loop.

12 Threads (15 minutes, 4 points)

```
class MyProgram extends Thread {
  static Object lock = new Object();
  int id;
  boolean pause;

  public MyProgram(int d) {
    this.id = d;
    this.pause = false;
  }
```

```
  public void setPause(boolean newPause) {
    this.pause = newPause;
  }

  public void run() {
    for(int i = 0; i < 4; i++) {
      synchronized(lock) {
        try {
          System.out.print(id);
          if(pause) lock.wait();
        } catch(InterruptedException ex) { }
      }
      sleep(500);
    }
  }

  public static void main(String[] args) {
    MyProgram[] prog = new MyProgram[3];
    for(int i = 0; i < prog.length; i++)
      (prog[i] = new MyProgram(i)).start();
    sleep(500);
    for(int i = 0; i < prog.length; i++)
      prog[i].setPause(true);
    System.out.print("@");
    synchronized(lock) {
      prog[0].setPause(false);
      lock.notifyAll();
    }
    try {
      for(int i = 0; i < prog.length; i++) prog[i].join();
    } catch(InterruptedException ex) {
      System.out.print("$");
    }
  }

  static void sleep(int time) {
    try {
      Thread.sleep(time);
    } catch(InterruptedException ex) {
      System.out.print("#");
    }
  }
}
```

Which of the following is a possible output of the above program

A 0120@12020121

B **02112@20100**

C 012012@012#012

D 012#000$

E 2012@01201201$

13 Failure to Terminate (5 minutes, 2 points)

When the code in Question 12 is run, it does produce output, but it never completes. This is because:

A The for loop at the beginning of main() starts the first thread, then waits for it to finish and is unable to start the next.

B When a thread executes lock.wait(), it is in the synchronized block on lock, and so main() cannot enter the synchronized block to run notifyAll().

C **When notifyAll() is called, threads other than prog[0] will call lock.wait() again.**

D When notifyAll() is called, it picks only one thread to leave the wait().

E Trick question, the program will complete normally.

14 Errors (5 minutes, 2 points)

Other than the problem in question 13, which of the following type of error would be most likely to occur with the code of question 12?

A Equivalence testing errors

B **Out-of-bounds errors**

C Scope errors

D Precision errors

Doubly Linked List

Questions 15 through 18 make use of the following code:

```
public class DLLNode<E> {
  // Single node of a doubly-linked list.
  private DLLNode previous;
  private E value;
  private DLLNode next;

  public DLLNode(E value) {
    previous = null;
    this.value = value;
    next = null;
  }

  private DLLNode(DLLNode previous, E value, DLLNode next) {
    this.previous = previous;
    this.value = value;
    this.next = next;
  }
```

```
  public DLLNode insertBefore(E value) {
    DLLNode newNode = new DLLNode<E>(previous, value, this);
    previous = newNode;
    return newNode;
  }

  public DLLNode insertAfter(E value) {
    DLLNode newNode = new DLLNode<E>(this, value, next);
    next = newNode;
    return newNode;
  }

  public DLLNode deleteGetAfter() {
    if (previous != null) previous.next = next;
    if (next != null) next.previous = previous;
    return next;
  }

  public DLLNode deleteGetBefore() {
    if (previous != null) previous.next = next;
    if (next != null) next.previous = previous;
    return previous;
  }

  public E getValue() {
    return value;
  }
}

public class Stack<E> {
  DLLNode top = null;

  public void push(E value) {
    if (top == null) top = new DLLNode<E>(value);
    else top = top.insertBefore(value);
  }

  public E pop() throws NullPointerException {
    if (top == null) throw new NullPointerException();
    else {
      E result = (E)top.getValue();
      top = top.deleteGetAfter();
      return result;
    }
  }
}
```

and the following code fragment (assume that variables exam1-5 are declared and instantiated; i.e., they are not null.)

```
  Stack<Exam> s = new Stack<Exam>();
  s.push(exam1);
  s.push(exam2);
```

```
s.pop().gradeIt();
s.push(exam3);
s.push(exam4);
s.push(exam5);
s.pop().gradeIt();
s.pop().gradeIt();
```


15 Stack Operations (5 minutes, 3 points)

In what order will the exams be graded given the above code fragment? (I.e., the order calls to .gradeIt() will occur)

A exam1 exam2 exam3 exam4 exam5

B exam5 exam4 exam3 exam2 exam1

C exam1 exam2 exam3

D exam2 exam4 exam3

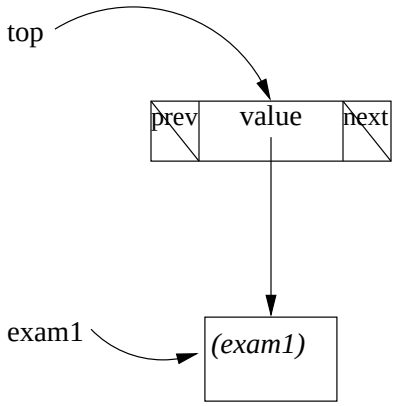
E **exam2 exam5 exam4**

Short Answer

Write your answers in the space provided. The space provided should be sufficient, but if you need more, use the back of the sheet.

16 Box and Pointer Diagrams (5 minutes, 5 points)

Below find a box-and-pointer diagram for the state of the data structure after the first two lines of the code fragment on page 9 (s has been created, s.push(exam1).) Complete the diagram showing the state of the data structure after the end of the code fragment (after the final s.pop().gradeIt();).



Added to the above diagram will be a second prev/value/next box, and an exam3 box. Top will point to the box pointing to exam 3, next will point to the box pointing to exam1, prev from that will point back to the exam 1.

Scoring: 1 for exam3, 1 for exam3 only, 1 for pointer 1 to 3, 1 for pointer 3 to 1, 1 for appropriate nulls

17 Stack Search (8 minutes, 4 points)

We would like to be able to search the stack, i.e., we want a method for Stacks as follows:

```
boolean search(E item); // Return true if and only if item is in the stack
```

Please write such a method for the class Stack on page 9. Note that it should not make any changes to the stack, only return true/false as appropriate.

Solution code:

```
class SearchableStack<E> extends Stack<E> {

  public boolean search(E item) {
    return recursiveSearch(item, top)
  }
  private boolean recursiveSearch(E item, DLLNode position) {
    if (position == null) return false;
    else if (position.getValue().equals(item)) return true;
    else return recursiveSearch(item, position.getNext());
  }
}
```

Scoring: 1 point for true if found, 1 for false if not found, 1 for getting termination right, 1 for false if empty list, 1 for recursive solution, 1 point for noting difficulty given lack of access to previous/next in DLLNode, 1 for writing as a class extending stack (up to 4).

18 Queue (10 minutes, 6 points)

We would like a queue similar to stack, except that the first item put in should be the first item to come out. It should have operations:

```
void enQueue(E item); // Put item at the end of the queue
E deQueue() throws NullPointerException(); // Get the first item in the queue.
```

Please give the code for such a class, using the DLLNode data structure.

A possible solution is:

```
public class Queue<E> {
  DLLNode head, tail;

  public void enQueue(E item) {
    if (tail == null) {
      DLLNode newNode = new DLLNode<E>(item);
      tail = newNode;
      head = newNode;
    } else
      tail.insertAfter(item);
  }

  public E deQueue() throws NullPointerException {
    if (head == null) throw new NullPointerException();
    else {
      E result = (E)head.getValue();
      head = head.deleteGetAfter();
```

```
      return result;
      }
  }
}
```


Scoring: 1 for correct class definition and methods, 1 for enQueue inserts, 1 for enQueue sets pointers correctly, 1 for deQueue gets right result, 1 for deQueue sets pointers correctly, 1 for proper deQueue exception.

19 I/O (4 minutes, 3 points)

A programmer wrote the following code to read an array of integers from a file.

```
String filename = "mydata";
DataInputStream stream = new DataInputStream(new FileInputStream(filename));
int[] data = new int[10000];
for(int i = 0; i < data.length; i++)
    data[i] = stream.readInt();
```

You notice it takes a long time to read the data from the file. Explain why it slow, and suggest a way to make it faster (for full credit, give the code to solve the problem.)

Data reading is not buffered, every read goes to the disk. Fix: change the first line to use BufferedInputStream, i.e., new DataInputStream(new BufferedInputStream(new FileInputStream(filename)));

Scoring: 1 point for buffered, 1 point for start at solution, 1 point for coded solution.

20 Pointers (10 minutes, 4 points)

```
class Node {
  public int value;
  public Node next;
}
```

Given a linked list of integers defined using the above class, write a **recursive** function that returns the maximum value in the linked-list. (A non-recursive solution will receive at most two points.)

The following is a possible solution:

```
int Max(Node head) {
  if(head == null)
    throw new NullPointerException("head is a null pointer.");
  else if(head.next == null)
    return head.value;
  else {
    int t = Max(head.next);
    if(t > head.value) return t;
    else return head.value;
  }
}
```

Scoring: 1 for proper base case, 1 for recursive case, 1 for proper calculation in recursive case, 1 for dealing with null head.

21 Synchronization (5 minutes, 2 points)


```
public class Chess extends Thread {

  private long id;

  public Chess(long id){
    this.id = id;
  }

  public void run(){
    move(id);
  }

    // INSERT FRAGMENT HERE
    System.out.print(id + " ");
    System.out.print(id + " ");
  }

  public static void main(String[] args) {
    Chess ch1 = new Chess(2);
    Chess ch2 = new Chess(4);
    ch1.start();
    ch2.start();
  }
}
```

Given the above and the following two code fragments:

```
1)  void move(long id) {
2)  synchronized void move(long id) {
```

Either give an example of output that is possible with fragment 1 but not possible with fragment 2, or explain why any possible result of fragment 1 is also possible with fragment 2.

A synchronized method will ensure that no other process is using the *current object*. In other words, when thread ch1 is started, no other thread can run the move method *for object ch1*. But since the second thread is running move for object ch2, any interleaving is possible - the synchronization has no effect on these two threads, so fragments 1 and 2 are essentially the same.

Scoring: 1 point for any result that interleaves 2s and 4s (good idea, but not correct), 2 points for pointing out that they synchronize on different objects, so synchronization has no effect.