CS1102s Data Structures and Algorithms

13/3/2009

Examination Questions Midterm 1

This examination question booklet has 7 pages, including this cover page, and contains 15 questions.

You have 35 minutes to complete the exam. Use a 2B pencil to fill up the provided MCQ form. Leave Section A blank. Fill up Sections B and C.

Java

Question 1: Consider the following Java program:

```
public class IntegerTest {
   public static Integer f(Integer i) {
            i = new Integer(2);
            return new Integer(3);
   }
   public static void main(String[] args) {
            Integer myInteger = new Integer(1);
            f(myInteger);
            System.out.println(myInteger);
   }
}
```

What will be the output of the program?

1 [A] 3

1 [B] 2

1 [C] 1

1 [D] Compilation error: return value of f is not used.

1 [E] None of the above

Answer 1:

1 [C] Remember that identifiers of object type are assigned to references to objects, and that each time a method/function is called, new locations are created for each parameter. Object references are copied to these locations. (In the JVM, this is done by creating a local variable area for each method call and by copying the arguments from the operand stack of the calling function invocation to the local variable area of the callee function invocation.)

Question 2: The type Integer is a subtype of the type Number. Consider the following Java program fragment:

```
class A {
  public void f(Number x) {
    System.out.println(x);
  }
}
Integer i = 4;
new A().f(i);
```

2 [A] This program leads to a runtime error because the function call f(i) does not find a function f that accepts arguments of type Integer.

2 [B] This program can be compiled with no error, and when executed, prints the number 4.

2 [C] This program leads to a compilation error because the the function call f(i) does not correspond to a function f in class A that accepts arguments of type Integer.

2 [D] This program leads to a compilation error because the **int** 4 is assigned to the Integer variable i.

2 [E] None of the above

Answer 2:

2 [B] An Integer ISA Number and thus, the function call matches the signature of f.

Question 3: Consider the following Java program fragment:

```
for (Animal a : myA) {...}
```

This fragment may be compiled without error if it is preceeded by the declaration

3 [A] none of the below

3 [B] java.util.Iterator myA

3 [C] Comparable myA

3 [D] java.util.Comparator myA

3 [E] java.util.Collection myA

Answer 3:

3 [E] Collections are Iterable and thus the iteration syntax can be used. None of the other classes/interfaces are Iterable, not even Iterator.

Question 4: The type Integer is a subtype of the type Number. Consider the following program fragment:

```
java.util.Collection<Number> c = new java.util.ArrayList<Integer>();
c.add(4);
```

4 [A] This program can be compiled with no error, and when executed, adds the number 4 to the Collection c.

4 [B] This program leads to a runtime error because the function call add(4) does not find a function add that accepts arguments of type **int**.

4 [C] This program leads to a compilation error because the the function call add(4) does not correspond to a function add in class Collection<Number> that accepts arguments of type Integer.

4 [D] This program leads to a compilation error because an ArrayList<Integer> object cannot be assigned to a variable of type Collection<Number>.

4 [E] None of the above

Answer 4:

4 [D] Remember the animal cages! An ArrayList<Integer> is not a subtype of ArrayList<Number>, and thus, the assignment is invalid.

Question 5: Which statement on tail recursion in Java is true:

5 [A] Tail recursion should be avoided in Java because the Java runtime system requires space on its runtime stack for each function call, and because tail recursion can be replaced by a loop that needs no extra space.

5 [B] Tail recursion should be used in Java whenever possible, because the Java compiler always optimizes it to a simple loop.

5 [C] Tail recursion should be used in Java, because the programs are easier to read than corresponding programs without loops and are just as efficient.

5 [D] Tail recursion should be avoided in Java, because programs with tail recursion can always be replaced by loops, which are typically easier to read.

5 [E] None of the above.

Answer 5:

5 [A] See page 90.

Program Analysis

Question 6: Consider the following Java program fragment:

```
for (int i=1; i < N; i = i * 3)
    System.out.println("Hello!");
```

Which one of the following statements about the runtime $R(N)$ is true?

6 [A] $R(N) = \Theta(\log N)$

6 [B] $R(N) = \Theta(N^3)$

6 [C] $R(N) = \Theta(\log^3 N)$

6 [D] $R(N) = \Theta(3^N)$

6 [E] $R(N) = \Theta(\sqrt[3]{N})$

Answer 6:

6 [A] The loop will be executed $\log_3 N$ times, which is $\Theta(\log N)$.

Question 7: Consider the following Java program fragment:

```
for (int i=1; i < N/2; i++)
    for (int j=0; j < N/4; j++)
        x = x + 1;
```

Which one of the following statements about the runtime $R(N)$ is true?

7 [A] $R(N) = O(N/8)$

7 [B] $R(N) = O(N)$

7 [C] $R(N) = O(N^2)$

7 [D] $R(N) = O(\log_8 N)$

7 [E] $R(N) = O(\sqrt[8]{N})$

Answer 7:

7 [C] Both $N/2$ and $N/4$ are $\Theta(N)$. Thus we have $R(N) = \Theta(N^2)$, and therefore $R(N) = O(N^2)$.

Question 8: Let F be a function such that $F(N) = \Theta(N^3)$. Which one of the following statements is true?

8 [A] $F(N) = O(N^2)$ holds

8 [B] $F(N) = O(N^2)$ does not hold

Answer 8:

8 [B] $O(...)$ gives an upper bound on the runtime.

Question 9: Let F be a function such that $F(N) = \Theta(N^3)$. Which one of the following statements is true?

9 [A] $F(N) = O(N^3)$ holds

9 [B] $F(N) = O(N^3)$ does not hold

Answer 9:

9 [A] See textbook.

Question 10: Let F be a function such that $F(N) = \Theta(N^3)$. Which one of the following statements is true?

10 [A] $F(N) = O(N^4)$ holds

10 [B] $F(N) = O(N^4)$ does not hold

Answer 10:

10 [A] $O(...)$ gives an upper bound on the runtime. $O(N^4)$ is not very precise, but valid.

Lists, Stacks, Queues

Question 11: Consider the following Java function:

```
public static <Any> void removeFirstHalf(java.util.ArrayList<Any> list) {
    int theSize = list.size() / 2;
    int i = 0;
    Iterator<Any> it = list.iterator();
    while (i++ < theSize) {
        it.next();
        it.remove();
    }
}
```

Which one of the following statements is true?

11 [A] Any call of the function on a non-empty list will encounter a runtime error because hasNext() is not called before next().

11 [B] Any call of the function on an empty list will encounter a runtime error because next() is called on the list in this case.

11 [C] Any execution of this program on a list of even size will remove the first half of the list in $O(N)$ where N is the size of the list.

11 [D] Any execution of this program on a list of even size will remove the first half of the list in $O(N^2)$ where N is the size of the list.

11 [E] None of the above

Answer 11:

11 [D] Quadratic complexity because remove will have to adjust the indices of all following entries, which takes linear time for each iteration through the while loop.

Question 12: Consider the following Java program fragment:

```
java.util.LinkedList<Integer> list = new java.util.LinkedList<Integer>();
for (int i = 0; i < N; i++)
    list.add(i,i);
for (int j=0; j <= N/2; j++)
    list.remove(N/2);
```

Which one of the following statements about the runtime $R(N)$ is true?

12 [A] $R(N) = \Theta(N + N/2)$

12 [B] $R(N) = \Theta(N)$

12 [C] $R(N) = \Theta(N^2)$

12 [D] $R(N) = \Theta(N^3)$

12 [E] None of the above

Answer 12:

12 [C] The first for loop is clearly linear.

The first iteration through the second for loop requires $N/2$ steps, to reach position $N/2$. The second iteration takes $N/2 - 1$, etc, until the $N/2 + 1$th iteration, which is immediate (if N is even), because the end of the list is reached. Note that LinkedList reaches a given position from the beginning or from the end, whichever way is shorter. Thus overall, we have a runtime of we have $\sum_{i=0}^{N/2} i = O(N^2)$.

Question 13: Consider the following Java program fragment:

```
java.util.ArrayList<Integer> list = new java.util.ArrayList<Integer>();
for (int i = 0; i < N; i++)
    list.add(i,i);
for (int j = N; j >= 0; j--)
    list.remove(j);
```

Which one of the following statements about the runtime $R(N)$ is true?

13 [A] $R(N) = \Theta(\log N)$

13 [B] $R(N) = \Theta(N)$

13 [C] $R(N) = \Theta(N^2)$

13 [D] $R(N) = \Theta(N^3)$

13 [E] None of the above

Answer 13:

13 [B] The first loop takes clearly linear time. The second loop immediately leads to a runtime error because the list only has N elements, with indices from 0 to $N-1$. Thus overall runtime $O(N)$.

Question 14: Consider the following Java program fragment, whose runtime R is a function of the input size N.

```
ArrayList<Integer> a = new ArrayList<Integer>();
for (int i = 0; i < 100; i++)
  a.add(0,i);
for (int j = 0; j < N; j++) {
  a.add(0,j);
  a.remove(0);
}
```

Which of the following is correct:

14 [A] $R(N) = \Theta(N)$

14 [B] $R(N) = \Theta(N^2)$

14 [C] $R(N) = \Theta(N^3)$

14 [D] $R(N) = \Theta(N \log N)$

14 [E] None of the above

Answer 14:

14 [A] First loop takes 100 steps, which is $O(1)$. Each iteration through the second loop needs to shift 100 elements to the right, and then 100 elements to the left. So each iteration through the second loop takes $O(1)$ time. Overall $O(N)$.

Question 15: Consider the following attempt at implementing a Queue data structure.

```
public class MyArrayQueue<T> extends java.util.ArrayList<T> {
    public void enqueue(T item) {
        add(size(),item);
    }
    public T dequeue() {
        return remove(0);
    }
}
```

Which one of the following statements is true?

15 [A] This class implements a stack instead of a queue, and is as efficient as the stack implementations given in the lectures.

15 [B] This class implements a stack instead of a queue, but is not as efficient as the stack implementations given in the lectures.

15 [C] This class implements the queue behavior correctly, and is as efficient as the solution given in the lectures.

15 [D] This class implements the queue behavior of LIFO correctly, but is not as efficient as the solution given in the lectures.

15 [E] This class implements the queue behavior of FIFO correctly, but is not as efficient as the solution given in the lectures.

Answer 15:

15 [E] The dequeue operation needs to shift all elements to the right, which requires a runtime proportional to the current size of the queue. The solution in the lectures requires only constant time for all queue operations.