CS 211 **Computers and Programming**

Fall 2004

Prelim I Solutions

10/14/2004

1. (Types, 10 points)

   Let $r1$ and $r2$ be variables that contain references to an object o. Answer the following questions.

   (a) (1 point) The type of $r1$ can be either a class type or an interface type. True or false? Explain briefly.

   (b) (1 point) The type of o can be either a class type or an interface type. True or false? Explain briefly.

   (c) (2 points) Which of the following statements is true about the types of $r1$ and o? No explanation needed.

      i. The type of $r1$ must be a super-type of the type of o.

      ii. The type of $r1$ must be a sub-type of the type of o.

      iii. There is no relationship in general between the type of $r1$ and the type of o.

   (d) (2 points) Which of the following statements is true about the types of $r1$ and $r2$? No explanation needed.

      i. The type of $r1$ must be a super-type of the type of $r2$.

      ii. The type of $r1$ must be a sub-type of the type of $r2$.

      iii. There is no relationship in general between the type of $r1$ and the type of $r2$.

   (e) For the rest of the questions, consider the downcasting expression `(Animal)r1`, where `Animal` is assumed to be a class in the Java program.

      i. (1 point) Which of the following statements is true? No explanation needed.

         A. Downcasting is always legal, so this operation will never throw an exception.

         B. Downcasting is always illegal, so this operation will always throw an exception.

         C. This downcasting operation will execute correctly only if the type of object o is a subtype of `Animal`.

      ii. (1 point) Which of the following statements is true? No explanation needed.

A. The type of object *o* remains unchanged by the downcasting operation.

B. The type of object *o* becomes `Animal` after the downcasting operation.

C. You do not have enough information to say whether the type of object *o* will be changed by the downcasting operation.

iii. (1 point) Which of the following statements is true? No explanation needed.

A. The type of reference `r1` becomes `Animal` after this downcasting operation.

B. The type of reference `r1` is unchanged after this downcasting operation.

C. You do not have enough information to say whether the type of `r1` will be changed by the downcasting operation.

iv. (1 point) Which of the following statements is true? No explanation needed.

A. The type of reference `r2` becomes `Animal` after this downcasting operation.

B. The type of reference `r2` is unchanged after this downcasting operation.

C. You do not have enough information to say whether the type of reference `r2` will be changed by the downcasting operation.

Answers:

Total: 10 points

(a) (1 point) True. The purpose of subtyping is to allow references to a general supertype. (No credit given for no explanation).

(b) (1 point) False. You can't instantiate an interface because the methods are not implemented. (No credit given without explanation).

(c) (2 points) The type of $r1$ must be a super-type of the type of o.

(d) (2 points) There is no relationship in general between the type of $r1$ and the type of $r2$.

(e) i. (1 point) This downcasting operation is legal only if the type of object o is a subtype of `Animal`.

ii. (1 point) The type of object o remains unchanged by the downcasting operation.

iii. (1 point) The type of reference $r1$ is unchanged after this downcasting operation.

iv. (1 point) The type of reference $r2$ is unchanged after this downcasting operation.

2. (Induction, 20 points)

   Use induction to prove the following results. Your answers must state clearly (i) the base case or cases, (ii) the inductive hypothesis, (iii) the inductive step, and (iv) the conclusion.

   (a) (10 points)
   $(1-\frac{1}{4})(1-\frac{1}{9})...(1-\frac{1}{n^2}) = \frac{n+1}{2n}$ for $n \geq 2$

   (b) (10 points)
   Show that
   $\frac{m!}{0!} + \frac{(m+1)!}{1!} + ... + \frac{(m+n)!}{n!} = \frac{(m+n+1)!}{n!(m+1)}$
   where m and n are non-negative integers. Hint: you can do an induction on either m or n, but the induction on n is easier.

   Answer:

   (a) (10 points)

   - (2 points) Base case: for n = 2, (1 - 1/4) = 3/4 = (2+1)/2*2 = 3/4.
   - (2 points) Inductive hypothesis: assume the result is true for some integer k $\geq$ 2.
   - (6 points) Inductive step:
     $(1-\frac{1}{4})(1-\frac{1}{9})...(1-\frac{1}{(k+1)^2})$
     $=$
     $(1-\frac{1}{4})(1-\frac{1}{9})...(1-\frac{1}{k^2})(1-\frac{1}{(k+1)^2})$
     $=$
     $\frac{(k+1)}{2k} * (1-\frac{1}{(k+1)^2})$
     $=$
     $\frac{(k+2)}{2(k+1)}$
     as desired.
   - (2 points) Conclusion:
     $(1-\frac{1}{4})(1-\frac{1}{9})...(1-\frac{1}{n^2}) = \frac{n+1}{2n}$ for $n \geq 2$

   (b) (10 points)

   - (2 points) Base case:
     $\frac{m!}{0!} = \frac{(m+1)!}{0!(m+1)} = \frac{m!}{0!}$

- (2 points) Inductive hypothesis: for some integer $k \geq 0$
  $\frac{m!}{0!} + \frac{(m+1)!}{1!} + ... + \frac{(m+k)!}{k!} = \frac{(m+k+1)!}{k!(m+1)}$
- (6 points) Inductive step:
  $\frac{m!}{0!} + \frac{(m+1)!}{1!} + ... + \frac{(m+k)!}{k!} + \frac{(m+k+1)!}{k+1!}$
  $= \frac{(m+k+1)!}{k!(m+1)} + \frac{(m+k+1)!}{(k+1)!}$
  $= \frac{(m+k+2)!}{(k+1)!(m+1)}$
- Conclusion:
  $\frac{m!}{0!} + \frac{(m+1)!}{1!} + ... + \frac{(m+n)!}{n!} = \frac{(m+n+1)!}{n!(m+1)}$ for all $m, n \geq 0$.

This page intentionally left blank.

3. (Recursion and Lists, 40 points)

   (a) (20 points) Write a *recursive* public class method that takes a list of Objects `lc` and a reference to an Object `o` as arguments, removes the first occurrence of `o`, if it exists, in `lc`, and returns a reference to the remaining list.

   The list `lc` is to be modified in place. Use the `equals` method to compare the object `o` with elements of the list `lc`.

   Your method must have the following signature.

   ```
   public static ListCell delete(ListCell l, Object o);
   ```

   The ListCell class is reproduced at the end of the exam for your convenience. You may NOT use the List class discussed in lecture. *No credit will be given if you create a new list; that is, if you allocate any new ListCell objects.*

   (b) (20 points) Write an *iterative* public class method that takes a doubly-linked list of Objects `dl` (constructed from the DLLCell class discussed in lecture) and a reference to an Object `o` as arguments, removes the first occurrence of `o`, if it exists, in `dl`, and returns a reference to the remaining list.

   The list `dl` is to be modified in place. Use the `equals` method to compare the object `o` with elements of the list `dl`.

   Your method must have the following signature.

   ```
   public static DLLCell delete(DLLCell dl, Object o);
   ```

   The DLLCell class is reproduced at the end of the exam for your convenience. *No credit will be given if you create a new list; that is, if you allocate any new DLLCell objects.*

(a) No credit given if new ListCell objects created.

```
public static ListCell delete(ListCell lc, Object o) {
  if (lc == null)
    return null;
                      //5 points for base case

  if (lc.getDatum().equals(o))
    return lc.getNext();
                      //  5 points for other base case
                      // -2 points for lc.datum
                      // -1 point  for ==
                      // -1 point  for accessing null pointer

  else {
    lc.setNext(delete(lc.getNext(), Object o));
    return lc;
  }
                      // 10 points for recursive case
                      // -3 for incorrect return value
                      // -5 for no return value
                      // -2 for incorrect setNext value
                      // -3 for incorrect recursive call
}
```

(b) You will probably get many solutions, so I suggest something like this.

```
Does code work correctly for null dl? 2 points

Does code work correctly if Object o does not occur in list dl? 2
points

If object o does occur in the list, are the next and previous
fields set correctly to splice cell out? 4 points

Are next and previous fields set correctly when cell being spliced
out is the first cell or the last cell? 4 points

Is the right list returned if the cell being spliced out is the
first cell? 4 points

Is the right list returned otherwise? 4 points
```

```
public static DLLCell delete(DLLCell dl, Object o) {
    //walk down list to find first occurrence of o if any
    DLLCell cursor = dl;
    while ((cursor != null) && !(cursor.getDatum().equals(o))) {
        cursor = cursor.getNext();
        }

    if (cursor == null)
        //did not find o in list
        return dl;
    //otherwise we found an occurrence of o
    //set fields of cells before and after cell at cursor
    DLLCell before = cursor.getPrevious();
    DLLCell after = cursor.getnext();
    if (before != null) before.setNext(cursor.getNext());
    if (after != null) after.setPrevious(cursor.getPrevious());

    if (cursor == dl)
        //cursor is at first cell, so get rid of first cell
        return dl.getNext();
    else return dl;
}
```

This page intentionally left blank.

4. (Trees and lists, 30 points)

   A *full* binary tree is a special case of a binary tree which has at least one node, and in which every node has either 0 or 2 children (i.e., no node can have just one child). For example, the tree in Figure 1(a) is a full binary tree while the binary tree in Figure 1(b) is not a full binary tree.

   (a) (15 points) Prove inductively that a full binary tree must have an odd number of nodes. Your proof must clearly state the base case, inductive hypothesis, inductive step and conclusion. Hint: use an induction on the height of the tree.

   (b) (15 points) Consider full binary trees where Integer values are stored in each tree cell. Draw all possible full binary trees whose pre-order traversal produces the following integer sequence: 10 12 9 22 18 76 16.(Hint: there is more than one such full binary tree).

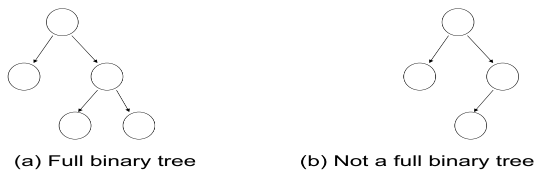


Figure 1: Full and non-full binary trees

Answer:

(a) (15 points) Note: 1 point was deducted for each missing label. A student who misunderstood the definition and claimed that a full binary tree has $\sum_{i=0}^{n} 2^i$ nodes received at most 7 points.

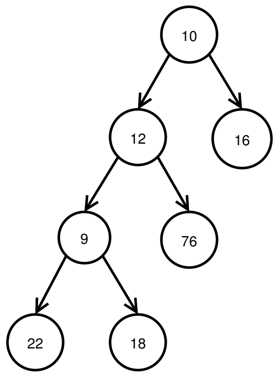
- (2 points) Base case: A full binary tree of height 0 has exactly one node, the root. Since one is an odd number, this proves the base case.
- (4 points) Inductive hypothesis: Assume that all full binary trees of height less than or equal to k (for some $k \geq 0$) have an odd number of nodes. (2 points for weak inductive hypothesis and 2 points for strong inductive hypothesis).
- (9 points) Inductive step. There were many different possible answers for this, one of which was given below. In all of them, 2 points were deducted for failing to explicitly use the inductive hypothesis. The remaining 7 points were allocated based on clarity and correctness.
  One possible inductive inductive step is given as follows:
  Consider a full binary tree of height $k + 1$.
  Since the height of the tree is greater than 0, the root has a non-empty left subtree or a non-empty right subtree.
  Since the tree is a full binary tree (either each node has 0 or 2 children), both the left and the right subtrees are non-empty.
  The left subtree is a full binary tree since it has at least one node (i.e., it is non-empty) and all nodes have either 0 or 2 children (since they are also nodes of the original full binary tree). Similarly, the right subtree is also a full binary tree.
  The left subtree has height at most k since the entire tree has height $k+1$ and the depth of the root of the left subtree is 1. Similarly, the height of the right subtree is at most k.
  By induction hypothesis, the left subtree and the right subtree each have an odd number of nodes.
  The total number of nodes in the tree is the sum of (1) the number of nodes in the left subtree of the root, (2) the number of nodes in the right subtree of the root, and (3) 1 (the root itself). Since (1) and (2) are odd, their sum is even, and adding 1 to this produces an odd number. Hence the number

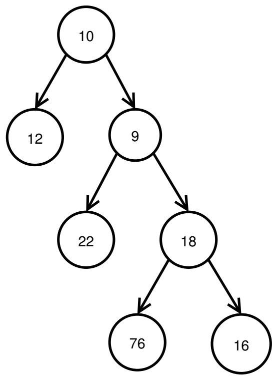
of nodes in the full binary tree of height $k + 1$ is odd.

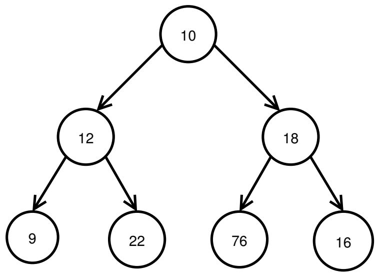
- Conclusion: A full binary tree has an odd number of nodes.

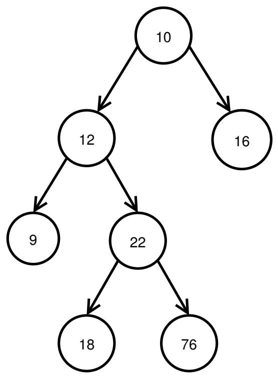
(b) (15 points)

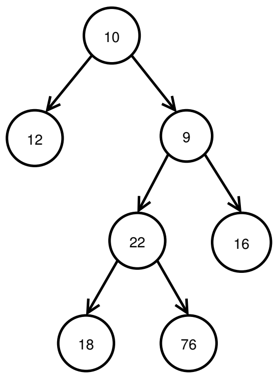
Three points were awarded for each correct tree. If more than five trees were drawn, then two points was deducted for each additional tree.











```
class ListCell {

 protected Object datum;
 protected ListCell next;

  public ListCell(Object o, ListCell n){
    datum = o;
    next = n;
  }

  public Object getDatum() {
    return datum;
  }

  public ListCell getNext(){
    return next;
  }

  public void setDatum(Object o) {
    datum = o;
  }

  public void setNext(ListCell l){
    next = l;
  }

 public String toString(){
   String rString = datum.toString();
   if (next == null) return rString;
   else return rString + " " + next.toString();
 }
}
```

```
class DLLCell {

 protected Object datum;
 protected DLLCell next;
 protected DLLCell previous;

  public DLLCell(Object o, DLLCell n, DLLCell p){
    datum = o;
    next = n;
    previous = p;
  }

  public DLLCell(Object o){
    datum = o;
  }

  public Object getDatum() {
    return datum;
  }

  public DLLCell getNext(){
    return next;
  }

  public DLLCell getPrevious(){
    return previous;
    }

  public void setDatum(Object o) {
    datum = o;
    }

  public void setNext(DLLCell dl) {
    next = dl;
    }

  public void setPrevious(DLLCell dl) {
    previous = dl;
    }
}
```

Course Feedback

You can detach this sheet from the exam and hand it to a TA to maintain confidentiality.

1. How many hours per week are you spending on the homework for this course?

2. Do you attend lecture regularly? What can we do to improve the lectures?

3. Do you attend section regularly? What can we do to improve section?

4. Do you see the consultants for assistance? What can we do to improve consulting?