COMP 2404B

Midterm Exam Solution -- Version 4

[out of 50 marks]

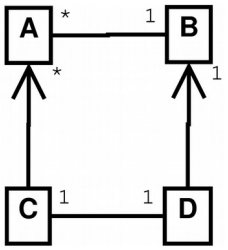
1. [8 marks]

```
// 2 marks for object Z
// 2 marks for object Y collection
class X { Y  objY[MAX]; Z objZ; };

// 1 mark for class definition
class Y { };

// 3 marks for object X
class Z { X objX[MAX]; };
```

2. [12 marks]



Grading:

- 2 marks: A-B directionality and multiplicity (1 mark each)
- 2 marks: B-A directionality and multiplicity (1 mark each)
- 2 marks: C-A directionality and multiplicity (1 mark each)
- 2 marks: C-D directionality and multiplicity (1 mark each)
- 2 marks: D-B directionality and multiplicity (1 mark each)
- 2 marks: D-C directionality and multiplicity (1 mark each)

Deductions: -2 marks for any additional association

3. [10 marks]

```
Book* List2::delFront() {
  Book* goner;
  Node* newHead;

// 2 marks for dealing with empty list case
// -- 1 mark for checking for empty list
// -- 1 mark for returning zero or NULL
  if (head == 0)
    return 0;

// 1 mark for only doing this if it's not an empty list
  goner = head->data;
  newHead = head->next;
// 2 marks for deallocating old head
  delete head;
// 2 marks for setting new head to correct value
  head = newHead;
// 1 mark for setting new head's prev to null
  newHead->prev = 0;

// 2 marks for returning correct book
// -- 0 out of 2 marks if node is deallocated before data is stored
  return goner;
}
```

4. [10 marks]

```
void List1::findAll(const string author, List1& list) {
  Node *currNode;
  currNode = head;

// 2 marks for correct loop over list (includes end condition and
// advancing currNode to next)
  while (currNode != 0) {

// 2 marks for comparing current book author and parameter
    if (currNode->data->getAuthor() == author) {

// 6 marks for adding the book to the list parameter
// -- 2 marks for calling addBack() or addFront()
// -- 2 marks for adding to list parameter
// -- 2 marks for using currNode data
// -- if existing add function not used, give max of 2 out of 6 marks for using
//    currNode data
      list.addBack(currNode->data);
    }
    currNode = currNode->next;
  }
}
```

5. [10 marks]

```
void Arr2::addFront(Book* b)
{
// 5 marks for the backward loop header (won't work in forward direction)
// -- 2 marks for starting at size
// -- 2 marks for ending at 1
// -- 1 mark for decrementing at every iteration
  for (int i=size; i>0; --i)

// 2 marks for moving each element one position towards the back of the array
    elements[i] = elements[i-1];

// 2 marks for setting elements[0] to the new book
// -- 0 out of 2 marks if b is dereferenced
  elements[0] = b;

// 1 mark for incrementing size
  ++size;
}
```