# COMP 2404A

## Midterm Exam Solution -- Version 2

**[out of 50 marks]**

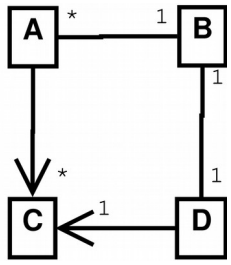
1. [8 marks]

```
// 3 marks for object Z
class X { Z objZ; };

// 2 marks for object X collection
class Y { X objX[MAX]; };

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

2. [12 marks]




Grading:

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

```
void List1::delFront(Book** b) {
  Node* oldHead;
// 3 marks for dealing with empty list case
// -- 1 mark for checking for empty list
// -- 1 mark for setting dereferenced b to zero or NULL
// -- 1 mark for returning
  if (head == 0) {
    *b = 0;
    return;
  }
// 1 mark for only doing this if it's not an empty list
  oldHead = head;

// 2 marks for setting new head
  head = head->next;

// 2 marks for setting dereferenced b (1 mark) to old head data (1 mark)
// -- 0 out of 2 marks if old head is deallocated first
  *b = oldHead->data;

// 2 marks for deallocating old head
  delete oldHead;
}
```

4. [10 marks]

```
void List1::findNewer(int year, 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 year and parameter
    if (currNode->data->getYear() > year) {

// 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 Arr1::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;
}
```