COMP 2404A

Midterm Exam Solution -- Version 1

[out of 50 marks]

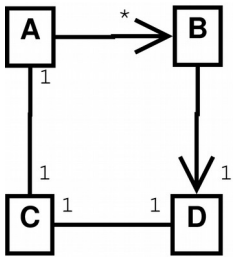
1. [8 marks]

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

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

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

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: C-A directionality and multiplicity (1 mark each)
- 2 marks: C-D directionality and multiplicity (1 mark each)
- 2 marks: B-D 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::addFront(Book* b)
{
// 4 marks for allocating and initializing new node
// -- 2 marks allocating node
// -- 1 mark initializing data
// -- 1 mark initializing next (this matters for the empty list case)
  Node* newNode = new Node;
  newNode->data = b;
  newNode->next = NULL;

// 3 marks for dealing with empty list case
// -- 1 mark for checking for empty list
// -- 2 marks for setting head and tail to new node (1 mark head, 1 mark tail)
  if (head == 0) {
    head = tail = newNode;
  }
// 3 marks for dealing with regular case
// -- 1 mark for only doing this if it's not an empty list
// -- 1 mark for setting new node's next to old head
// -- 1 mark for setting head to new node
  else {
    newNode->next = head;
    head = newNode;
  }
}
```

4. [10 marks]

```
void List1::find(int year, Book** b) {
  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) {
// 4 marks for setting b parameter and returning
// -- 2 marks for using dereferenced b as destination
// -- 1 mark for using current node data as source
// -- 1 mark for returning here
      *b = currNode->data;
      return;
    }
    currNode = currNode->next;
  }
// 2 marks for setting dereferenced b parameter to null if book not found
  *b = NULL;
}
```

5. [10 marks]

```
bool Arr1::delFront(Book& b)
{
// 2 marks for dealing with empty array case
  if (size == 0)
    return false;

// 2 marks for setting b to first element
// -- 0 out of 2 marks if b is dereferenced
  b = elements[0];

// 2 marks for forward loop header (must loop over entire array)
// 2 marks for moving each element one position towards the front of the array
  for (int i=0; i<size; ++i)
    elements[i] = elements[i+1];

// 1 mark for decrementing size
  --size;

// 1 mark for returning true
  return true;
}
```