COMP 2404B

Midterm Exam Solution -- Version 3

[out of 50 marks]

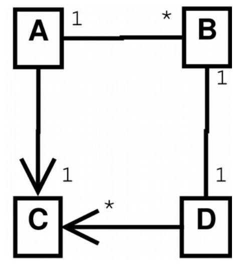
1. [8 marks]

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

// 3 marks for object X
class Y { X objX; };

// 1 mark for class definition
class Z { };
```

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 List2::addFront(Book* b)
 {
 // 5 marks for allocating and initializing new node
 // -- 2 marks allocating node
 // -- 1 mark initializing data
 // -- 1 mark initializing prev
 // -- 1 mark initializing next (this matters for the empty list case)
   Node* newNode;
   newNode = new Node;
   newNode->data = b;
   newNode->prev = 0;
   newNode->next = 0;

 // 1 mark for setting new node’s next to current head
   newNode->next = head;

 // 2 marks for setting current head’s prev to new node, assuming non-empty list
   if (head != 0)
     head->prev = newNode;

 // 2 marks for setting head to new node
   head = newNode;
 }
```

4. [10 marks]

```
void List2::findOldest(Book** b) {
  Node *currNode;
  int  oldest = 3000;

// 3 marks for initializing dereferenced b to zero or NULL in case list is empty
  *b = 0;

  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 the current oldest book
    if (currNode->data->getYear() < oldest) {
      oldest = currNode->data->getYear();

// 3 marks for setting dereferenced b to current oldest book
      *b = currNode->data;
    }
    currNode = currNode->next;
  }
}
```

5. [10 marks]

```
void Arr2::delFront(Book** b)
{
// 3 marks for dealing with empty array case
// -- 1 mark for checking for empty array
// -- 1 mark for setting dereferenced b to zero or NULL
// -- 1 mark for returning here
  if (size == 0) {
    *b = 0;
    return;
  }

// 2 marks for setting b to first element
// -- 0 out of 2 marks if b is not dereferenced
  *b = elements[0];

// 4 marks for shifting remaining elements
// -- 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;
}
```