Midterm 2 Information

- The second midterm will be on Friday, 1 Apr 2005, from 1:10–2:00 p.m.
- The exam will primarily cover lectures from Monday, 14 Feb 2005 through Wednesday, 23 Mar 2005, as well as the associated readings and assignments for those topics. (please see the class schedule for exact coverage)
- This exam is closed book and no calculating devices of any type will be allowed. You are allowed, however, to prepare in advance *the back of this page* with whatever notes you wish to place on it, and you may use this page during the exam. When the exam is over, staple this sheet to the rest of your exam.

  Furthermore, we will provide you with fresh copies of the second and third page of this handout for your use on the day of the exam. Therefore, when preparing your notes on the back of this page, you need not worry about including material from the second page.

ListADT

```
template <typename Object>
class List {

public:
  class Position {
  public:
    Object& element() const throw(InvalidPositionException);
    bool isNull() const;
    }
  };

  int size() const;
  bool isEmpty() const;
  Position first() const throw(EmptyContainerException);
  Position last() const throw(EmptyContainerException);
  bool isFirst(const Position& p) const throw(InvalidPositionException);
  bool isLast(const Position& p) const throw(InvalidPositionException);
  Position before(const Position& p) const
    throw(BoundaryViolationException, InvalidPositionException);
  Position after(const Position& p) const
    throw(BoundaryViolationException, InvalidPositionException);
  Position insertFirst(const Object& element);
  Position insertLast(const Object& element);
  Position insertBefore(const Position& p, const Object& element)
    throw(InvalidPositionException);
  Position insertAfter(const Position& p, const Object& element)
    throw(InvalidPositionException);
  void remove(const Position& p) throw(InvalidPositionException);
  void replaceElement(const Position& p, const Object& element)
    throw(InvalidPositionException);
  void replaceElement(const Position& p, const Position& q)
    throw(InvalidPositionException);
};
```

VectorADT

```
template <typename Object>
class Vector {
public:
  int size() const;
  bool isEmpty() const;
  Object& elemAtRank(int r) const;
  void replaceAtRank(int r, const Object& e);
  void removeAtRank(int r);
  void insertAtRank(int r, const Object& e);
};
```

SequenceADT

```
template <typename Object>
class Sequence : public List<Object>, public Vector<Object> {
public:
  Position atRand(int rank) const throw(BoundaryViolationException);
  int rankOf(Position position) const throw(InvalidPositionException);
};
```