Name: . . . . . .

ID Number: . . . . . .

COMP103 Summer: Midterm

18th January 2006.

Instructions

- Examination Time: **90 minutes**.
- There are 90 marks in total.
- Guideline: spend approximately 1 minute per mark.
- Answer **all** the questions.
- Write your answers in the boxes in this test paper and hand in all sheets.
- Every box with a heavy outline requires an answer.
- If you do not understand a question, ask for clarification.
- There is useful documentation in an appendix at the end of the exam paper.

Total: (90 marks)	**Marks**		
1. Collections	[10]	1	
2. Analysing cost of Algorithms	[11]	2	
3. Iterators	[17]	3	
4. Implementation of Collections	[16]	4	
5. Searching	[15]	5	
6. Sorting Algorithms	[21]	6	
		Total:	

Question 1. Collections [10 marks]

ADTs in java are specified by an interface. For each of the following five methods, explain its operation and the ADT (interface) to which it belongs:

i. `public E push(E item){...}`

ii. `public E poll(){...}`

iii. `public V put(K key, V value){...}`

iv. `public E pop(){...}`

v. `public E offer(E item){...}`

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 2. Algorithmic Costs [11 marks]

(a) [5 marks]

In the following code fragment, assume that `data` is an array of `int`, and the length of `data` is *n*. What is the aysmptotic cost of the following loop, expressed in terms of *n*?

```
for(int i = 1; i < data.length; i+=2){
  for(int j=i-1; j < i; j++)
    System.out.print(data[j] + ", ");
  System.out.print(data[i] + ", ");
}
```

(b) [6 marks]

Complete the following table:

Algorithm	Best	Worst	Stable
Selection Sort		$O(n^2)$	
Merge Sort			yes
Quick Sort	$O(nlog(n))$		

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 3. Iterators [17 marks]

(a) [2 marks] Briefly state the purpose of an Iterator.

(b) [6 marks] Complete the following code that displays the contents of a Christmas stocking. The appendix includes the Set ADT for your reference.

```
Set<String> christmasStocking;
christmasStocking = new SetImp<String>("toy car", "ball", "sweets");

Iterator<Set> stocking =

while(
     system.out.println(
);
```

(c) [6 marks] Repeat the program, but use the *for each* loop instead.

```
Set<String> christmasStocking;
christmasStocking = new SetImp<String>("toy car", "ball", "sweets");

for(
     system.out.println(
```

(d) [3 marks] In the `christmasStocking` code above, what determines the order in which the contents are printed to the screen

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 4. Interfaces, Abstraction and Implementations [16 marks]

(a) [2 marks] Briefly state the purpose of an abstract class.

(b) [2 marks] State one way in which an abstract class is like an interface.

(c) [2 marks] State one way in which an abstract class is **not** like an interface.

(d) [10 marks] Given the following `AbstractHuman` class for representing information about people:

```
public abstract class AbstractHuman{
  protected String name;
  protected int DoB;

  public abstract String getGender(){}  // return a string identifying gender

  public String getName(){              // return string name
    return name;
  }

  public int getDoB(){                 // return date of birth
    return DoB;
  }
}
```

Complete the following concrete `Female` subclass of `AbstractHuman` that represents information about female humans. Note: only implement methods required from inheriting `AbstractHuman`.

```
public class Female
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 5. Binary Search [15 marks]

(a) [5 marks] Suppose an array contained the values:

anglia	classic	cortina	escort	fiesta	granada	laser	mondeo	prefect	taunus	zephyr
0	1	2	3	4	5	6	7	8	9	10

(i.)[2] Show the sequence of values from this vector that binary search would look at if it were searching for the value “granada”.

(ii.)[3] Show the sequence of values from this vector that binary search would look at if it were searching for the value “classic”.

(b) [10 marks] The code below implements the add method for a set using a linear search in a sorted array. The method searches through the array until it locates the correct position to insert the new item. If the item is already in the set, then the method doesn't do anything and returns `false`. If the item is not in already the set, the remaining elements in the array are moved and the new item inserted and the method returns `true`.

Modify this code to use the binary seach method provided. Note: You do not need to provide an implementation of binary search.

```
public class SortedArray<T> implements Set<T>{
private T[] data;
private int count=0; // number of elements in the Set
private static int initialCapacity = 10;
private Comparator c;

public SortedArray(Comparator comparator){
  c = comparator;
  data = (T[])new Object[initialCapacity];
}

public boolean add(T item){
  int scan;

  if (item == null) throw new NullPointerException();

  for(scan = 0; scan < count-1; scan++){
    int result = c.compare(item, data[scan]);
    if(result == 0)
      return false;
    else if(result > 0)
      break;
  }
  ensureCapacity();
  for (int i=count; i > scan; i--)
    data[i]=data[i-1];
  data[scan]= item;
  count++;
}

private int binarySearch(T item){
  // Implements binary search (do not write this)
  // It returns the position in the array where the
  // item ought to be.
  ...
}

:
.
```

Question 6. Sorting Algorithms [21 marks]

(a) [2 marks] What idea do the merge and quicksort algorthims use to achieve their good performance?

(b) [4 marks] What is the principle difference in design between Merge Sort and Quicksort algorithms?

(c) [15 marks] Suppose an `array` containing the following values is to be sorted using Quicksort.

prefect	taunus	cortina	granada	anglia	escort	mondeo	zephyr
0	1	2	3	4	5	6	7

Our version of Quicksort uses a partition algorithm (as described in class) that selects the first element as the pivot. Show the state of the `array` at the end of the first three calls to partition. If needed, see the appendix for the full quicksort implementation (for sorting arrays of ints).

after 1:

0	1	2	3	4	5	6	7

after 2:

0	1	2	3	4	5	6	7

after 3:

0	1	2	3	4	5	6	7

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

****

Appendix

```
public interface Collection<E>{
  public boolean isEmpty();
  public in size();
  public Iterator<E> iterator();
}

public interface Set extends Collection<E>{
  public boolean contains(E element);
  public void add(E element);
  public void remove(E element);
}


class QuickSort {

    void sort(int a[], int lo0, int hi0) throws Exception {
        int lo = lo0;
        int hi = hi0;
        if (lo >= hi)
          return;
        else if( lo == hi - 1 ) {
            /*
             *  sort a two element list by swapping if necessary
             */
            if (a[lo] > a[hi])
              swap(a,lo,hi);
            return;
        }


        /*
         *  Pick a pivot and split
         */

        int pivot = partition(a, lo, hi);

        /*
         *  Recursive calls, elements a[lo0] to a[pivot-1] are less than or
         *  equal to pivot, elements a[pivot+1] to a[hi0] are greater than
         *  pivot.
         */

        sort(a, lo0, pivot-1);
        sort(a, pivot+1, hi0);
    }

    void sort(int a[]) throws Exception {
        sort(a, 0, a.length-1);
    }
```

```
    int partition(int data[], int lo, int hi)throws Exception{
      int pivot = data[lo];
      int scan = lo+1;
      int mark = scan;

      while(scan <= hi){
        if (data[scan] < pivot){
          swap(data, scan, mark);
          mark++;
        }
        scan++;
      }

        swap(data, lo, mark-1);
        return mark-1;
    }


    void swap(int data[], int a, int b) throws Exception{
        int T = data[a];
        data[a] = data[b];
        data[b] = T;
    }
}
```