THE UNIVERSITY OF WESTERN AUSTRALIA

FACULTY OF
Engineering, Computing
and Mathematics

School of Computer Science & Software Engineering

MID-SEMESTER TEST April 2015

DATA STRUCTURES AND ALGORITHMS 2200 (CITS2200)

SURNAME: _____ STUDENT NO: _____

GIVEN NAMES: _____ SIGNATURE: _____

There are 10 questions and you have 35 minutes to complete the test. Each question has exactly one correct answer.

(1)	A	B	C	D
(2)	A	B	C	D
(3)	A	B	C	D
(4)	A	B	C	D
(5)	A	B	C	D
(6)	A	B	C	D
(7)	A	B	C	D
(8)	A	B	C	D
(9)	A	B	C	D
(10)	A	B	C	D

Good luck

For each of the following items, please enter one answer A, B, C or D, on the sheet provided.

1. Assume that `Labradoodle` has been defined as a subclass of `Dog`, and `Pound` has been defined as a stack of `Dog`s (that is, it has the same operations as a generic Stack, but it consists of items of type `Dog` rather than type `Object`). Assume that the code

```
Pound p = new Pound();
Dog d = new Dog();
Labradoodle l = new Labradoodle();
```

   is directly followed by one of the following:

   i.
```
p.push(l);
l = p.pop();
```

   ii.
```
p.push(l);
d = p.pop();
```

   iii.
```
p.push(d);
l = p.pop();
```

   iv.
```
p.push(l);
l = (Labradoodle) p.pop();
```

   Identify which of the above will cause an error?

   (a) (i) and (iii)
   (b) (ii) and (iii)
   (c) (iii) only
   (d) (i), (iii) and (iv)

2. Suppose that $f(n)$ is $O(g(n))$, $g(n)$ is $O(h(n))$ but $h(n)$ is not $O(f(n))$. Which of the following are possible functions for f, g and h?

   (a) $f(n) = 2^n$, $g(n) = n2^{(n)}$ and $h(n) = 2^{n+4}$.
   (b) $f(n) = 8n$, $g(n) = 2n$ and $h(n) = n + \sqrt{n}$.
   (c) $f(n) = n^3$, $g(n) = 3n^2$ and $h(n) = 6n$.
   (d) $f(n) = n^2$, $g(n) = n^2 + n$ and $h(n) = n^2 \lg n$.

3. The following code is for the Partition method used by the QUICKSORT algorithm:

```
procedure PARTITION(A, p, r)
  x ← A[r]; i ← p − 1
  for j ← p to r − 1
    do if A[j] ≤ x
      then i ← i + 1
        exchange A[i] ↔ A[j]
  exchange A[i + 1] ↔ A[r]
  return i + 1
```

Suppose that PARTITION$(A, 1, 6)$ is called over the array $A = [8, 4, 2, 7, 1, 5]$ (assuming the array indexes from 1). What is the result?

(a) $A = [4, 2, 1, 5, 8, 7]$ and 4 is returned.

(b) $A = [1, 2, 4, 5, 7, 8]$ and 5 is returned.

(c) $A = [4, 2, 1, 5, 8, 7]$ and 5 is returned.

(d) $A = [4, 2, 1, 5, 7, 8]$ and 4 is returned.

4. A `deque` (double-ended queue) is implemented using an array called `items` and left and right indices called `left` (an index to the leftmost item) and `right` (an index to the rightmost item) respectively. The deque is cyclic (or "wraps around") so that all space in the array can be used.

The method `pushLeft` adds an item to the left end of the deque and is implemented as follows:

```
public void pushLeft(char c) throws Overflow {
  if (!isFull()) {

    << missing code >>

  }
  else throw new Overflow(‘‘Pushing to full deque’’.)
}
```

Which of the following is a correct implementation of the missing lines:

(a)
```
left = (left-1)%items.length;
items[left] = c;
```

(b)
```
left = (left+1)%(items.length-1);
items[left] = c;
```

(c)
```
left = left-1;
if (left=right+1) left = (left-1)%items.length;
items[left] = c;
```

(d)
```
left = left-1;
if (left == -1) left = items.length-1;
items[left] = c;
```

5. A (singly) linked implementation of a Queue contains the following instance variables:

   `front` — a reference to the front of the queue, that is, the end with the item that has been in longest
   `back` — a reference to the back of the queue, that is, the end with the item that was added most recently

   The `enqueue` method can be implemented as follows:

```
public void enqueue (Object a) {
  if (isEmpty()) {
    front = new Link(a,null);
    back = front;
  }

    << missing code >>

}
```

   All operations in the queue must be able to operate in constant time.

   Which of the following is a correct implementation of the missing code?

(a)
```
else front = new Link(a,front);
```

(b)
```
else back.successor = new Link(a,null);
```

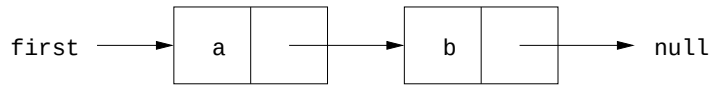
(c)
```
else {
  front.successor = new Link(a,front);
  front = front.successor;
}
```

(d)
```
else {
  back.successor = new Link(a,null);
  back = back.successor;
}
```

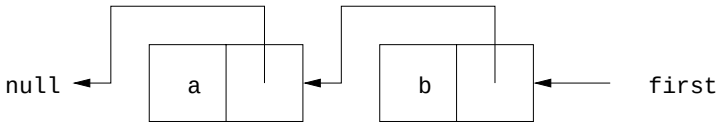
6. Let `Link` be an object with two member variables:

```
public class Link {
  char item;
  Link successor;
}
```

Assume that `first` is a reference to a `Link` object containing 'a', whose successor is a link object containing 'b', whose successor is `null`.



Which of the following successfully reverses this structure as follows?



(a)
```
first.successor.successor = first;
first = first.successor;
first.successor = null;
```

(b)
```
first.successor = first;
first = first.successor;
first.successor.successor = null;
```

(c)
```
Link temp = first.successor;
temp.successor = first;
first.successor = null;
first = temp;
```

(d)
```
Link temp = first;
first = temp.successor;
first.successor = temp;
```

7. A block implementation of a List contains the following instance variables:

   `block` — an array of objects that stores the items in the list
   `before` — a reference to the before-first position
   `after` — a reference to the after-last position

   It is used with a window class that contains one variable:

   `index` — the position in the list of the window item

   The `insertBefore` method can be implemented as follows:

```
public void insertBefore (Object e, WindowBlock w) throws
  OutOfBounds, Overflow {
      if (!isFull()) {
    if (!isBeforeFirst(w)) {

            << missing code >>

        }
      else throw new OutOfBounds ("Inserting before start of list.");
  }
      else throw new Overflow("Inserting in full list.");
```

   Which of the following is the best implementation of the missing code?

   (a)
```
for (int i=block.length-1; i>=w.index; i--)  block[i+1] = block[i];
after++;
block[w.index] = e;
w.index++;
```

   (b)
```
for (int i=after-1; i>=w.index; i--)  block[i+1] = block[i];
w.index++;
block[w.index] = e;
after++;
```

   (c)
```
for (int i=after-1; i>=w.index; i--)  block[i+1] = block[i];
after++;
block[w.index] = e;
w.index++;
```

   (d)
```
for (int i=w.index; i<block.length; i++)  block[i+1] = block[i];
after++;
block[w.index] = e;
w.index++;
```

8. The following method searches an array (stored in `block`) to see if the same item appears twice.

```
public boolean hasMatch (int[] block) {
     boolean found = false;
     for (int i=0; i<block.length; i++) {
          for (int j=0; j<block.length; j++)
               found = found || (i != j && block[i]==block[j]);
     }
     return found;
}
```

   If the function $f(x)$ describes the time performance of this method, where x denotes the size of the block, which of the following is the smallest 'big O' for $f(x)$?

   (a) $f(x)$ is $O(1)$
   (b) $f(x)$ is $O(\log n)$
   (c) $f(x)$ is $O(n)$
   (d) $f(x)$ is $O(n^2)$

9. A mountain climber can climb 10 kms in a day. She can descend 30 kms in a day. What is her amortized rate of travel?

   (a) 10 kms per day
   (b) 15 kms per day
   (c) 20 kms per day
   (d) 30 kms per day

10. Suppose that you have a block implementation of a stack of objects, and you would like to call a method, `removeNulls` that removes all of the null elements in that stack.

```
public class Stack{
        public Object[] items;
        public int top;

        public Stack(int size){
                items = new Object[size];
                top = -1;
        }
                //push, pop, peek and isEmpty are implemented as in the notes

        public void removeNulls(){
                //missing code
        }
}
```

Which of the following is an incorrect implementation of the missing code in the `removeNulls` method?

(a)
```
for(int i = 0; i< top; i++){
        if(items[i] == null){
                for(int j = i; j < top; j++) items[j] = items[j+1];
                top--;
        }
}
```

(b)
```
int count = 0;
for(int i = 0; i< top; i++){
        if(items[i] != null) items[count++] = items[i];
}
top = count;
```

(c)
```
Stack s = new Stack();
while(!isEmpty()){
        Object o = pop();
        if(o != null) s.push(o);
}
while(!s.isEmpty()) push(s.pop());
```

(d)
```
Stack s = new Stack();
while(!isEmpty()){
        Object o = pop();
        if(o != null) s.push(o);
}
items = s.items;
top = s.top;
```