# The University of British Columbia

## Computer Science 260

## Sample Midterm Examination

## Term 2, Winter 2011

**Instructor: K. S. Booth** **Time: 30 minutes (half an hour)** **Total marks: 39**

First ___________________________ Last ____________________________ Student No _____________
**Printed first name** **Printed last name**

Signature ________________________________ Lecture Section **201** Lab Section ____________

## ☛ This examination has 6 pages. Check that you have a complete paper.

This is a closed book exam. Notes, books or other materials are not allowed.

Answer all the questions on this paper. The marks for each question are given in { braces }. Use this to manage your time.

Good luck.

**READ AND OBSERVE THE FOLLOWING RULES:**

1. Each candidate should be prepared to produce, upon request, his or her Library/AMS card.
2. No candidate shall be permitted to enter the examination room after the expiration of 30 minutes, or to leave during the first 30 minutes of the examination.
3. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors in examination questions.
4. **CAUTION** — Candidates guilty of any of the following, or similar dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

    a. Making use of any books, papers or memoranda, calculators or computers, audio or visual cassette players, or other memory aid devices, other than those authorized by the examiners.
    b. Speaking or communicating with other candidates.
    c. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

| Page | Marks | Max |
|---|---|---|
| 3 | | 10 |
| 4 | | 8 |
| 5 | | 2 |
| 6 | | 5 |
| **Total** | | **25** |

## 1. Multiple choice questions { 10 marks — 1 mark per question }

On the next page is a series of short fill-in-the-blanks questions. All of your answers are to be selected from the list below. You may find it convenient to remove this page from the answer booklet so you can look at it while you answer the questions that follow.

1) abstract
2) accessor
3) asymptotic
4) base
5) big-Oh
6) binding
7) BST
8) complexity
9) concrete
10) constructor
11) contiguous
12) dangling
13) deep
14) depth
15) derived
16) destructor
17) dimension
18) driver
19) dynamic
20) encapsulation
21) executable
22) extent
23) FIFO
24) function
25) global
26) header
27) heap
28) heterogeneous
29) homogeneous
30) inheritance
31) inorder
32) instance
33) invariant
34) leaf
35) leak
36) level
37) lifetime
38) LIFO
39) linear
40) linked
41) member
42) memory
43) module
44) node
45) object
46) operator
47) overloading
48) path
49) pointer
50) polymorphism
51) postorder
52) preorder
53) private
54) protected
55) public
56) quadratic
57) queue
58) record
59) recursion
60) reference
61) root
62) scope
63) segment
64) shallow
65) source
66) stack
67) static
68) string
69) struct
70) subclass
71) superclass
72) synthesize
73) this
74) traverse
75) type
76) value
77) v-pointer
78) v-table
79) virtual
80) visibility

Each statement will have one ■■■■■■■■■ within it, which is where the missing term or phrase would appear. Choose the best answer from among those above and write its number in the space provided in the first column. Do not write the term or phrase. It may be a good idea to read over the list of terms and phrases before you start answering. Some of the terms listed may not appear in any of the statements, some may appear in more than one statement, and some many appear in exactly one statement.

**Continue on to the next page…**

**You may remove this page from the exam booklet.**

Read the instructions on the previous page. Enter the number for your answer in the first column. Do not write words!

| | | |
|---|---|---|
| **(a)** | | Variables whose extent (lifetime) is ■■■■■■■■ are created using the **`new`** operator. |
| **(b)** | | Variables that are created using the **`new`** operator are stored in the ■■■■■■■■ memory segment. |
| **(c)** | | Automatic variables are stored in the ■■■■■■■■ memory segment. |
| **(d)** | | Global variables are stored in the ■■■■■■■■ memory segment. |
| **(e)** | | If memory is not returned after it is no longer being used, a memory ■■■■■■■■ occurs. |
| **(f)** | | The zeroth or implicit parameter in a method invocation is associated with the identifier ■■■■■■■■. |
| **(g)** | | The default parameter passing mechanism for objects is call-by-■■■■■■■■. |
| **(h)** | | Call-by-■■■■■■■■ if often used when large objects are passed as parameters. |
| **(i)** | | A compiler will ■■■■■■■■ an assignment operator if one is not declared for a class. |
| **(j)** | | A class is the same as a struct except that member functions and member variables are ■■■■■■■■ instead of public unless explicitly declared otherwise. |

## 2. The Big Three { 5 marks }

**(a) { 3 marks }** Consider the following definitions for the copy helper method and the cleanup helper method for a class **`Ralph`** whose declaration is not shown.

```
Ralph::Ralph( const Ralph &other ) {
    copy( other );
}

Ralph::~Ralph() {
    cleanup();
}
```

In the space below, write a complete definition for the overloaded assignment operator for the **`Ralph`** class using the standard conventions described in lecture.

**ANSWER:**

**(b) { 2 marks }** Consider the following client function that uses the **`Ralph`** class.

```
void function( int n ) {
    Ralph Z;
    Ralph W = Z;
}
```

For each of the "big three" methods, fill in the second column with the number of times that each "big three" method is invoked when the client function **`function()`** is executed once. Be sure to count all of the invocations of the "big three" methods. At least one is invoked more than once.

| "Big Three" method name | Number of times method is invoked |
|---|---|
| `copy constructor` | |
| `destructor` | |
| `overloaded assignment operator` | |

## 4. Run-time and stack frames { 6 marks }

Answer the two parts to this question that follow, one on this page and one on the next page of the exam. Each refers to the C++ program shown below.

```
#include <iostream>


using namespace std;


void foo( int n ) {
  static int m = 0;
  cout << m << " " << n << endl;
  m = n*n;
  int x = m + n;
}


int  main() {
  int k = 6;
  foo( ++k );
  foo( k++ );
}
```

**(a) { 2 marks }** In the space below, write the output that would be produced if the program above were executed. You may assume that the program compiles and links without errors.

**ANSWER:**

**Continue on to the next page…**

**Name:** **Student No:**

**(b) { 4 marks }** Refer to the program on the previous page when answering this part of the question.

In the table below, draw a diagram of the runtime stack immediately after the function **`foo()`** has been called for the second time but before any statements within the body have been executed.

- For each entry in the stack frames associated with **`main()`** and **`foo()`** provide a label to the left of the stack frame entry and indicate the value stored in the location within the entry.
- Use the symbol "?" to indicate an undefined or unspecified value, but if you think you know what the value will be based on your knowledge of how the stack is managed, enter the actual value.
- Draw a double line above the top of the stack frame for **`foo()`** similar to the double line that is already above the stack frame for **`main()`**.
- Assume that no local variables exist in either function other than those that are explicitly declared.

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| **`foo()`** | | |
| | | |
| | | |
| **`main()`** | **`k`** | |
| | **`return value`** | & |
| | **`return address`** | * |