MARKING KEY **The University of British Columbia** MARKING KEY

Computer Science 252

1st Midterm Exam

1:00 PM, Tuesday, October 12, 2004

Instructors: K. Booth & N. Hutchinson **Time: 45 minutes** **Total marks: 45**

Name_____Student No_____
(PRINT) (Last) (First)

Signature_____Lecture Section _____

☛ **This examination has 10 pages. Check that you have a complete paper.**

This is a closed book exam. Notes, books or other materials are not allowed, but you may use **one** piece of 8-1/2 x 11 paper (both sides) with your own notes either hand-written, photocopied, or both.

Answer all the questions on this paper. Give **short but precise** answers. Always use point form where it is appropriate. 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 twenty minutes, or to leave during the first twenty minutes of the examination.

3. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities 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	8.58	10
4	7.33	10
6	6.62	8
7	4.32	7
9	4.53	10
Extra	1.00	1
Total	**33.36**	**45**

1. Multiple choice questions { 20 marks -- 10 marks per page }

On the next two pages there are a series of short fill-in-the-blanks statements. All of your answers are to be selected from the list below.

1)	accessor	12)	directory	23)	parameterized constructor
2)	assignment operator	13)	executable	24)	precedence
3)	constructor	14)	file	25)	precociousness
4)	convert constructor	15)	g++	26)	pwd
5)	copy cat	16)	header	27)	shallow
6)	copy constructor	17)	include	28)	source
7)	core	18)	interface	29)	title
8)	current directory	19)	make	30)	validation
9)	deep	20)	mkdir	31)	verification
10)	default constructor	21)	mutator	32)	WYSIWYG
11)	destructor	22)	object		

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)	**12**	A student in lab section L2M should have a [redacted] with the following pathname in which all of the files to be submitted for Assignment 1 are placed. `~/cs252/a1-oct01.m`
(b)	**29**	For every assignment a [redacted] file that has the name(s) and student ID number(s) of the student(s) submitting the assignment must be submitted electronically.
(c)	**19**	For every assignment a [redacted] file that can be used to compile and link the various source and object files must be submitted electronically.
(d)	**13**	The result of compiling, linking and loading is the [redacted] file for a program.
(e)	**7**	On Unix a file with the name [redacted] is created when there is a bug in your program that causes the program to exit abnormally. It is a good idea to remove these files on a regular basis because they take up a lot of disk space.
(f)	**16**	Normally the declarations for a C++ class are in the [redacted] file for the class.
(g)	**28**	Normally the definitions for a C++ class are in the [redacted] file for the class.
(h)	**17** (1) **16** (1/2)	Often the file with the declaration for a C++ class will specify one or more system [redacted] files that declare classes used by that class.
(i)	**18**	The declarations for a C++ file specify the [redacted] to the class, and are the only information about the class available to the client code that uses the class.
(j)	**24**	The [redacted] of an operator in C++ determines when it will be evaluated within an expression if there are no parentheses that specify the order of evaluation.

(continued on the next page)

(continued from previous page)

(k)	**6**	The recommended style for the [redacted] method in a C++ class is to simply invoke the **`copy()`** helper function.
(l)	**11**	The recommended style for the [redacted] method in a C++ class is to simply invoke the **`clear()`** helper function.
(m)	**2**	The recommended style for the [redacted] method in a C++ class is to check for a special case (in which case nothing need be done but return immediately) or else invoke the **`clear()`** helper function and then invoke the **`copy()`** helper function and return.
(n)	**1**	[redacted] methods always have the keyword **`const`** at the end of their declaration because they may read the values of member variables but they never change those values.
(o)	**21**	[redacted] methods never have the keyword **`const`** at the end of their declaration because they may change the values of member variables.
(p)	**10**	[redacted] methods never have parameters and they always have the same name as the class to which they belong.
(q)	**23**	[redacted] methods may have multiple parameters (one, two, or many) and they always have the same name as the class to which they belong.
(r)	**6**	[redacted] methods always have one parameter whose type is the class itself and they always have the same name as the class to which they belong.
(s)	**4**	[redacted] methods always have one parameter whose type is some other class or built-in type and they always have the same name as the class to which they belong.
(t)	**6** (1) **2** (1/2)	The compiler always provides a component-wise [redacted] for a C++ class unless one is declared for the class, and this is used to initialize new objects in the class for the code: `SomeClass Object1 = Object2;`

Below is a declaration for the **`Queue`** class from a **header file, `Queue.h`** from Assignment 1, followed by the definition for the **`enqueue()`** method from the **source file, `Queue.cpp`**. These will be useful when you answer the next two questions on this exam.

header file

```
typedef int Item_type;                  // type for items in the Queue

class Queue
{
   public:
      Queue();
      bool enqueue( const Item_type& item );
      Item_type dequeue();
      bool empty() const;
      bool full() const;
      void print() const;
   private:
      static const int CAPACITY = 50;   // maximum number of items
      int count;                        // number of items in the Queue
      int front;                        // index of first item
      Item_type data[CAPACITY];         // array of items
};
```

source file

```
bool Queue::enqueue( const Item_type& item )
// Add one item at the rear of the Queue
// PRE:  (none)
// POST: if the Queue is not full the item is added and TRUE is returned
//       otherwise the Queue is not changed and FALSE is returned
{
   if ( full() ) return false;          // can't add to a full Queue
   if ( (front+count) >= (CAPACITY) )
   {
      for (int i=0; i<count; i++) data[i]=data[front+i];
      front=0;
   };
   data[front+(count++)] = item;
   return true;
}
```

Continue on to the next page…

You may remove this page from the exam booklet.

2. Writing source code for methods { 8 marks }

(a) **{ 4 marks }** Write a complete definition for the **`empty()`** method as it might appear in the source file for the **`Queue`** class. Be sure to include appropriate preconditions and postconditions.

ANSWER:

```
bool Queue::empty() const
// Determine if Queue has no items in it

// PRE:  none     NOTE: this is required on all future exams!

// POST: The Queue is unchanged.  The return value indicates
// the status of the Queue (TRUE if empty, FALSE otherwise).
{
   return count <= 0;    // <= instead of == is safer!
}
```

(a) **{ 4 marks }** Write a complete definition for the default constructor method as it might appear in the source file for the **`Queue`** class. Be sure to include appropriate preconditions and postconditions.

ANSWER:

```
Queue::Queue()
// Default constructor

// PRE: none     NOTE: this is required on all future exams!

// POST: A new Queue is created with no items in it.
{
   count = 0;
   front = 0;            // 0 because 1st enqueue() stores in data[front]
}
```

3. Exception handling { 7 marks }

Write a complete definition for the **`dequeue()`** method that throws an exception (a **`logic_error`**) containing the message **`"Queue is empty"`** as it might appear in the source file for the **`Queue`** class. Be sure to include appropriate preconditions and postconditions. You may assume that any shifting of items in the array will be done by the **`enqueue()`** method, so you do not have to do any shifting in **`dequeue()`**.

ANSWER:

```
Item_type Queue::dequeue()  throw logic_error     // "throw" optional

// Remove one item from the front of the Queue

// PRE:  Queue must not be empty      NOTE: this is required

// POST: An item is removed from the Queue and returned, unless the
// Queue is empty, in which case an exception is thrown.

{
   if ( count < 1 ) throw logic_error( "Queue is empty" );
   count--;
   return data[front++];
}
```

4. Dynamic storage allocation, pointers, and compiler errors { 10 marks }

Here are four similar but different C++ code sequences that use static and dynamic arrays of characters and pointers. These will be referred to on the next page. Differences in the code sequences from the original **Sequence 1** are underlined (the underlining does not affect compilation, it is only there for your convenience in identifying the differences).

```
// Sequence 1 ================================================
  char alpha[27] = "abcdefghijklmnopqrstuvwxyz";
  char* p = NULL;
  char* q = new char[2];
  q[0]='A'+1;
  q[1]=*(alpha+5);
  *(p = alpha) = 'C';
  cout << q[0] << q[1] << " " << alpha << endl;

// Sequence 2 ================================================
  char alpha[27] = "abcdefghijklmnopqrstuvwxyz";
  char* p = NULL;
  char* q = new char[2];
  q[0]='A'+1;
  q[1]=*(alpha+5);
  *(alpha = p) = 'C';
  cout << q[0] << q[1] << " " << alpha << endl;

// Sequence 3 ================================================
  char alpha[27] = "abcdefghijklmnopqrstuvwxyz";
  char* const p = NULL;
  char* q = new char[2];
  q[0]='A'+1;
  q[1]=*(alpha+5);
  *(p = alpha) = 'C';
  cout << q[0] << q[1] << " " << alpha << endl;

// Sequence 4 ================================================
  const char alpha[27] = "abcdefghijklmnopqrstuvwxyz";
  char* p = NULL;
  char* q = new char[2];
  q[0]='A'+1;
  q[1]=*(alpha+5);
  *(p = alpha) = 'C';
  cout << q[0] << q[1] << " " << alpha << endl;
```

Continue on to the next page…

You may remove this page from the exam booklet.

(a) **{ 8 marks }** For each of the four code sequences on the previous page write one of two things:

If the code compiles without an error message, write the single line of output that results from executing the code sequence.

Otherwise: (1) write the word "ERROR"; (2) write the single line of source code in which the compiler first detects an error in the code sequence; and (3) write a brief description of the error explaining why it is an error. You may ignore all errors on subsequent lines in each code sequence after the first error in each code sequence.

ANSWERS:

```
Sequence 1: Bf Cbcdefghijklmnopqrstuvwxyz


Sequence 2: ERROR  *(alpha = p) = 'C';  // cannot assign to array


Sequence 3: ERROR  *(p = alpha) = 'C';  // cannot change const pointer


Sequence 4: ERROR  *(p = alpha) = 'C';  // cannot discard const
```

(b) **{ 1 mark }** During what phase of the compilation process would the following error message be produced by the g++ compiler on the undergrad Unix servers?

```
Undefined       first referenced
 symbol             in file
main            /cs/local/lib/pkg/gcc-3.2/lib/gcc-lib/sparc-sun-
                    solaris2.9/3.2/crt1.o
ld: fatal: Symbol referencing errors. No output written to a.out
collect2: ld returned 1 exit status
```

ANSWER:

The message is generated in the loader (linker) phase of compilation.

(c) **{ 1 mark }** What causes this particular error?

ANSWER:

The function main() is missing, probably because there is no driver in any of the object files.

Continue your answers here – make sure to identify the question(s) whose answer(s) are here!