MARKING KEY

The University of British Columbia

MARKING KEY

Computer Science 260

Midterm #1 Examination

12:30 noon, Tuesday, February 14, 2012

Instructor: K. S. Booth **Time: 70 minutes (one hour ten minutes)** **Total marks: 70**

First _____ Last _____ Student No _____

Printed first name **Printed last name**

Signature _____ Lecture Section **201** Lab Section _____

☛ **This examination has 11 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 10 minutes, or to leave during the first 10 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	Mean	Max
3	7.36	10
4	7.23	10
5	5.71	10
6	5.75	8
7	3.57	5
8	3.10	7
9	7.22	10
10	3.48	10
Total	43.42	**70**

1. Multiple choice questions { 30 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)	accessor	17)	executable	33)	lifetime	49)	scope
2)	automatic	18)	extent	34)	linker	50)	selection
3)	binding	19)	field	35)	member	51)	shallow
4)	block	20)	frame	36)	memory	52)	signature
5)	boolean	21)	function	37)	module	53)	source
6)	compiler	22)	global	38)	mutator	54)	stack
7)	conditional	23)	header	39)	object	55)	static
8)	constructor	24)	heap	40)	operator	56)	string
9)	contiguous	25)	heterogeneous	41)	overloading	57)	struct
10)	dangling	26)	homogeneous	42)	pointer	58)	synthesize
11)	deep	27)	implicit	43)	private	59)	this
12)	depth	28)	initialization	44)	public	60)	type
13)	destructor	29)	instance	45)	record	61)	value
14)	driver	30)	invariant	46)	recursion	62)	visibility
15)	dynamic	31)	leak	47)	reference	63)	void
16)	encapsulation	32)	level	48)	relational	64)	while

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.

Name: **Student No:**

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

(a)	**33**	A synonym for the term "extent" is [redacted]. It refers to the duration during which a variable exists in memory. lifetime
(b)	**49**	A synonym for the term "visiblity" is [redacted]. It refers to the portion of a program in which a particular identifier is bound to a particular variable. scope
(c)	**2**	[redacted] variables are always stored in the stack memory segment. automatic
(d)	**15**	[redacted] variables are always stored in the heap memory segment. dynamic
(e)	**22**	[redacted] variables are visible to the linker and are always stored in the static memory segment. global
(f)	**55**	[redacted] variables are not visible to the linker and are always stored in the static memory segment. static
(g)	**47**	The call-by-[redacted] parameter passing mechanism does not exist in the C programming language. reference
(h)	**47**	The call-by-[redacted] parameter passing mechanism is always used in a copy constructor in the C++ programming language. reference
(i)	**43**	Member variables in a class are usually declared to be [redacted]. private
(j)	**44**	A struct is the same as a class except that member variables are by default [redacted]. public

(This question is continued on the next page.)

Put a slash through every wrong answer, and an X through every blank answer. Total the number of correct answers and write at top of page AND on first page.

(This question is continued from the previous page.)

(k)	**58**	A compiler will [redacted] a destructor if one is not declared for a class. synthesize
(l)	**20**	The memory in the stack segment that is allocated at run time when a function or method is invoked is called a stack [redacted]. frame
(m)	**27**	The pointer variable **`this`** that is "built in" to every method is the [redacted] parameter for the method. implicit
(n)	**25**	Unlike arrays, structs are [redacted] data types. heterogeneous
(o)	**63**	The keyword [redacted] is used to indicate that a function does not return a value. void

(p)	**47**	`int i;`
		`int &k = i;`
		The variable **`k`** is a [redacted] because of the "**`&`**".
		reference

(q)	**52**	The number and the types of the parameters passed to a function are part of the [redacted] for the function. signature
(r)	**28**	The process of [redacted] uses syntax that is similar to an assignment, but does not perform an assignment. initialization
(s)	**11**	A [redacted] copy usually allocates new dynamic memory for one or more of the member variables in an object. deep
(t)	**51**	A [redacted] copy usually does <u>not</u> allocate new dynamic memory for any of the member variables in an object. shallow

(This question is continued on the next page.)

(This question is continued from the previous page.)

(u)	**5**	There are two binary logical or [redacted] operators in C and C++, "**`&&`**" and "**`||`**". boolean
(v)	**48**	The binary operator "**`==`**" is a [redacted] operator in C and C++. relational
(w)	**7**	The three-part operator "***`value1 ? value2 : value3`***" is the [redacted] operator in C and C++. conditional
(x)	**64**	There are two variants of the [redacted] control flow, one that tests before each iteration of a loop and one that tests after each iteration. while
(y)	**4**	A sequence of statements enclosed in curly braces "**`{`**" and "**`}`**" is a [redacted]. block
(z)	**23**	The [redacted] file for a class contains the class declaration. header
(aa)	**53**	The [redacted] file for a class contains the class definitions. source
(bb)	**34**	The [redacted] will report an error if a program defines the same function more than once. linker
(cc)	**6**	The [redacted] will report an error if a program fails to declare a variable that is used in an assignment statement. compiler
(dd)	**56**	The [redacted] type does not exist in C but is available in C++ as one of the standard classes that extend the basic language. It is often a better choice than the "array of **`char`**" type in C. string

2. Class declarations and method definitions { 13 marks }

Refer to the following UML (Universal Modeling Language) for both parts of this question. It specifies a simplified variation of the **Date** class that was used in Labs 2 and 3.

Date
- year_: int
- month_: string
- day_: int
+ Date()
+ setDate(year: int, month: string, day: int): void
+ toString() const: string
- intToString(n: int): string

(a) { 8 marks } Write a complete class header file for the **Date** class that matches the UML diagram.

ANSWER:

```
#ifndef _DATE_H_
#define _DATE_H_

#include <string>

using namespace std;

class Date {
  private:
    int year_;
    string month_;
    int day_;
    static string intToString( int n);
  public:
    Date();
    void setDate( int year, string month, int day );
    string toString() const;
};

#endif
```

½ mark off for each part that is incorrect
- #ifndef, #define, #endif
- #include, using
- class, class name
- private, public
- year_, month_, day_
- static, string return type, name inToString, parameter
- constructor
- void, method name, parameters
- string return type, method name, const

½ mark off for "extra" stuff ("const", etc.)

Continue on to the next page…

(b) { 5 marks } Write a complete definition for the **`toString()`** method in the **`Date`** class.

The format for the text representation of a **`Date`** should be "**`Month dd, yyyy`**" where the **`Month`** is the name of the month spelled out (not a number), **`dd`** is the day in the month (a one- or two-digit number), and **`yyyy`** is the year (a number).

You may assume that all of the other methods have been implemented and that all of the member variables have appropriate values.

ANSWER:

```
string Date::toString() const
{
  return month_ + " "
       + intToString( day_ ) + ", "
       + intToString( year_ )
}
```

½ mark off for each part that is incorrect
return type, class name, method name, no parameters, const
month_ does not need conversion
blank, convert day, comma, convert year

½ mark off for "extra" stuff ("const", etc.)

3. Class declarations and method definitions with dynamic memory { 27 marks }

Refer to the following class declaration for all parts of this question.

```
class IntSet {
  private:
    static const int CAPACITY = 5;
    int capacity_;
    int *data_;
    int count_;

    . . .

  public:
    IntSet( int capacity=CAPACITY )
      : capacity_(capacity), data_(new int[capacity]), count_(0) {}
    ~IntSet();
    IntSet( const IntSet &other );
    IntSet& operator=( const IntSet &other );
    void insert( int entry );
    void remove( int entry );
    bool find( int entry );
    int get( int index );
};
```

The return type for insert() was "int" on the exam for the declaration, but should have been "void" (which is what it was for the definition) so it has been changed here. It made no difference for the question.

(a) { 2 marks } How many distinct signatures are declared for constructors in the class declaration?

ANSWER:

3

All or nothing.

There are TWO constructors, default and parameterized in the first method plus the copy constructor (total of THREE).

(b) { 5 marks } Using the conventions we adopted in lecture, write the declarations for the two helper methods that should be where the "`. . .`" appear in the class declaration above for the **`IntSet`** class.

ANSWER:

```
void copy( const IntSet &other );

void cleanup();
```

1/2 mark off for each part that is incorrect
return type, method name, const, type, reference, parameter name
return type, method name, no parameters

½ mark off for extra stuff ("const", etc.)

Continue on to the next page…

Consider the following code example that uses the **`IntSet`** class declared on the previous page. The code is a definition for the **`insert()`** method as it might appear in the **`IntSet.cpp`** file.

```
void IntSet::insert( int entry )
{
   if ( find( entry ) ) return;
   if ( count_ < capacity_ )
   {
      data_[ count_ ] = entry;
      count_++;
   }
}
```

(c) { 10 marks } Write a complete definition for the **`remove()`** method for the **`IntSet`** class based on the class declaration on page 8 and the implementation described in lecture, which is very similar to the discussion in the textbook in Chapter 6 about how to remove items from a **`vector`**. The key idea is to overwrite the entry that is being removed with the last entry and then decrease the size by one.

ANSWER:

```
void IntSet::remove( int entry )
{
  for ( int i=0; i<count_; i++ )
  {
    if ( data_[i] == entry )
    {
      count_--;
      data_[i] = data_[count_];
      return;
    }
  }
  return;
}
```

1 mark off for each part that is incorrect
- return type, class name, method name, parameter type, parameter name
- iteration (for or while), only look at "count" number of entries
- test to find entry
- decrement count
- move last entry to vacant entry

1 mark off for "extra" stuff ("const", etc.)

Both of the returns are optional. The first return could be a "break" instead.

Continue on to the next page…

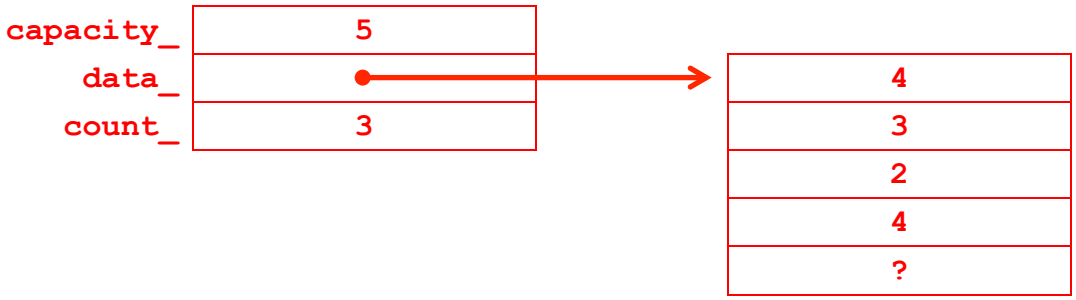
The next piece of code is a definition for **`main()`** as it might appear in the **`IntSet-driver.cpp`** file.

```
int main()
{
   IntSet A;
   for ( int k=1; k<5; k++ ) A.insert( k );
   A.remove( 1 );
   A.remove( 2 );
   A.insert( 2 );
}
```

(d) { 10 marks } Draw a diagram of the memory associated with the **`IntSet`** object **`A`** after the above code has executed. You should use the implementation we described in lecture for the **`remove()`** method. The code you wrote in part (c) does this, but you should be able to answer this part of the question even if you did not complete part (c).

Use the symbol "**?**" to indicate an undefined or unspecified value. Label each of the member variables.

ANSWER:



1 mark off for each entry that is incorrect (including the arrow)

1 mark off for each extra entry in either the object or the array

1 mark off for each missing or incorrect label

TOTAL cannot be negative

Continue your answers here – make sure to identify the questions whose answers are here!

Additional marking instructions

[Instructions will be provided prior to the marking session.]