CPSC 200 Fall 2001
Midterm I—03 October 2001

$Name(Printed)$: _____

$Signature$: _____

$StudentNumber$: _____

- Write the word circled above on each page of your exam. Do not put any other identifying marks on any page of your exam. Failure to put the circled word on a page of your exam may result in no marks being awarded for that page.
- *Read each question carefully. Ask yourself what the point of the question is. Check to make sure that you have answered the question asked.*
- This is a **50** minute exam. This exam contains **8** pages of questions not including this cover page. Make sure that you have all of them.
- Answer all questions on the exam sheet. If you do some of your work on the back of a page, clearly indicate to the marker what work corresponds with which question.
- Partial marks shall be awarded for clearly identified work.
- This exam counts as **20%** of your total grade. There are **50** points total on the exam.

ACRE	AREA	BALE	BAND	BARD	BASS
BETA	BIRD	BLOT	BOOK	BREW	CAMP
CHIP	CLAN	COAT	COIL	CORN	CROW
CURL	DARK	DEER	DOSE	DROP	DUCK
DUSK	FARE	FILM	FLAX	GAZE	GIFT
GOLD	GULF	HINT	HORN	HULL	IBOU
INCH	IRIS	ISLE	KERN	KILN	KITE
LANE	LARK	LENS	LOFT	LURE	MALT
MANX	MESH	MINK	MOTH	MOVE	MUSK
NAVY	NEWT	NOON	OATS	OBOE	OPAL
PARK	PINE	POET	RAFT	REED	RING
RUBY	RUFF	SEAM	SEED	SHOP	SILK
SINE	SNIP	SOAP	STUB	TASK	TAXI
TEAM	TELL	TEXT	TIDE	TILT	TOIL
TOME	TOUR	TURN	VANE	VISA	WALL
WICK	WOLF	WRIT	YARD		

Sums

$$\begin{aligned}
\sum_{i=1}^{n} i &= n(n+1)/2, \\
\sum_{i=1}^{n} i^2 &= (n+1)^3/3 - (n+1)^2/2 + (n+1)/6, \\
\sum_{i=1}^{n} i^3 &= (n+1)^4/4 - (n+1)^3/2 + (n+1)^2/4 = \left(\sum_{i=1}^{n} i\right)^2, \\
\sum_{i=1}^{n} 2^i &= 2^{n+1} - 1.
\end{aligned}$$

Logarithms and Exponential

$$a^x = y \qquad \Leftrightarrow \qquad x = \log_a y$$

$$\log a^s b^t = s \log a + t \log b, \qquad (x^a)^s \cdot \left(x^b\right)^t = x^{as+bt}$$

in particular $\log ab = \log a + \log b$, $\log a/b = \log a - \log b$, $\log a^s = s \log a$. To change from logarithms base a to logarithms base b use

$$\log_b x = \frac{\log_a x}{\log_a b}.$$

Factorials

$$n! = n \cdot (n-1)! = \prod_{i=1}^{n} i \qquad\qquad 5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

True-False

Circle the *best* answer. No marks shall be awarded for questions where it is not clear which answer is circled. If you provide a reason for your answer it may count for part marks.

(1) **1.** One of the $o(N \log N)$ algorithms for the maximum-subsequence-sum problem uses recursion. **TRUE FALSE**

(1) **2.** If $f(n) = o(g(n))$ then it is necessarily true that $f(n) = O(g(n))$. **TRUE FALSE**

(1) **3.** $n! = o(n^e)$ **TRUE FALSE**

(1) **4.** In order to be able to produce machine-code to call a non-templated function the compiler must be able to see the function definition (body). **TRUE FALSE**

(1) **5.** $n! = o(n^n)$ **TRUE FALSE**

Templates

6. Normal software development practice when using C++ is to separate each module into its interface and implementation parts, which are then stored in separate `.h`- and `.cpp`-file. The `.cpp` are then compiled to create an object file. Some current C++ compilers cannot handle templates that are written this way.

(2) (a) Give one strategy for dealing with templates in `.h` and `.cpp` files that works with most compilers.

(2) (b) Why does your strategy work?

(3) **7.** (a) Write a templated `findMinimum` function that takes a `vector` of objects and returns the value of the smallest item.

(1) (b) Someone who uses your faunction to test whether a `vector<char*>` object is sorted will likely be surprised by the result. Why?

(2) (c) Show how to fix this problem.

Asymptotic Analysis

8. Consider the following functions of n

- 1.05^n,
- n^2,
- $n^{(7/6)}$,
- $3n - 2$,
- $n + 5$,
- $\log n$,
- $n \log n$, and
- $\binom{n}{2}$

(3) (a) List these in order of growth, from slowest growing to fastest growing.

(1) (b) Indicate which of these are "Θ"-equivalent?

(1) (c) Where would $\log_2 n!$ fit into the above list?

(3) **9.** Write down the formal definitions for:

(a) $f(n) = \mathrm{o}(g(n))$

(b) $f(n) = \mathrm{O}(g(n))$

(5) **10.** Use the formal definition of $g(n) = \Omega(f(n))$ and mathematical induction to show that $n! = \Omega(5^n)$. [*Hint:* What would make a good n_0?]

(4) **11.** Suppose that $\lim_{n\to\infty} f(n)/g(n)$ either exists or goes to $+\infty$. Match up the following statements:

A.	$f(n) = \mathrm{O}(g(n))$	(a)	$\lim_{n\to\infty} f(n)/g(n) = 0$
B.	$f(n) = \Omega(g(n))$	(b)	$0 \le \lim_{n\to\infty} f(n)/g(n) < +\infty$
C.	$f(n) = \Theta(g(n))$	(c)	$0 < \lim_{n\to\infty} f(n)/g(n) < +\infty$
D.	$f(n) = \mathrm{o}(g(n))$	(d)	$0 < \lim_{n\to\infty} f(n)/g(n) \le +\infty$

A. matches ____. **B.** matches ____. **C.** matches ____.

D. matches ____.

(4) **12.** How many times longer do you expect it to take to sort 1000000 elements than 1000 elements

(a) when using a $\Theta(n \log n)$ method (like heap-sort),

(b) when using a $\Theta(n^{(4/3)})$ method (like a variant of Shell sort).

13. The function shown in Figure 1 takes a *sorted* array of *positive* doubles called `data` of size `size`, and returns `true` if and only if the sum of the entries in `data` is greater than or equal to 1.

(2) (a) Give a Θ-estimate of the *worst-case* running time of this code.

(2) (b) Give a Θ-estimate of the *best-case* running time of your algorithm.

```
bool sumAtLeast1(double data[], int size)
{
    double goal = 1.0 ;
    int i = 0 ;
    while (i<size && goal>(size-i)*data[i])
        {
        goal -= data[i] ;
        ++i ;
        }
    return (i<size) ;
}
```

Figure 1: Code for question 13

(2) (c) Write down a useful loop invariant for the loop in this code.

(2) (d) Write down a useful loop *variant* for the loop in this code.

14. The code shown in Figure 2 gives a recursive algorithm to search an ordered `vector` of double values for the index of the smallest location whose value exceeds `value`.

(2) (a) By examining the code which seems like the most likely equation for the running time $T(n)$ of `upper_bound(vec,value,0,n)`?

i. $T(n) = c_1 + c_2 n + 2T(n/2)$

ii. $T(n) = c_1 + c_2 n + T(n/2)$

iii. $T(n) = c_1 + 2T(n/2)$

iv. $T(n) = c_1 + T(n/2)$

Circle the roman numeral of your choice. Justify your decision.

```
#include "upper_bound.h"
// upper_bound assumes that vec is sorted in increasing order
//  returns the smallest I, lower<=I<high such that value<vec[I]
//  returns high if no such I exists.  We presume high<=vec.size().
int
upper_bound(vector<double> const& vec, double value,
            int lower, int high)
{
    if (lower==high)
        return high ;
    int middle = (lower+high) /2 ;
    int answer ;
    if (vec[middle]>value)
        answer = upper_bound(vec, value, lower, middle) ;
    else
        answer = upper_bound(vec,value, middle+1,high) ;
    return answer ;
}
```

Figure 2: Code for question 14-b.

(4) (b) Use telescoping sums to find a Θ-estimate for $T(n)$

Question	Score
1	/1
2	/1
3	/1
4	/1
5	/1
6	/4
7	/6
8	/5
9	/3
10	/5
11	/4
12	/4
13	/8
14	/6
Total	/50