Name	
Student number	

CMPT 225
Spring 2017
T. Shermer

Midterm Examination.

Please write your answers on the question sheets. No notes, books, or electronic devices of any sort allowed. Write clearly. This exam is scheduled for 50 minutes.

Question 1 (30 points total; 3 each)

Short answer questions. Complete sentences are not required. Justification for your answer is not required.

(a) What are three C++ loop constructs (statement types)?

do, for, while

(b) In C++, what is the difference between `--a` and `a--` ?
the first decrements a, then gives the decremented value of a.
the second decrements a but gives the value before the decrement.

(c) True or false? Any function that uses **`new`** in C++ is required to use **`delete`**.
False. Many functions use new without using delete (any function that returns a new object, for instance.) It is *recommended* that code that uses **new** has a corresponding **delete, but not at the function level.**

(d) If a C++ function has the statement/declaration:

```
Donkey eeyore;
```

Does one of Donkey's constructors get called, and if so, which one?

Yes. The default one.

(e) What is the worst-case time complexity of *push* for a stack implemented as a (growable) array?
O(n)

(f) What is the average-case time complexity of of *push* for a stack implemented as a (growable) array?
O(1)

(g) What is the worst-case time complexity of of *push* for a stack implemented as a linked list?

O(1)

(h) What is the average-case time complexity of of *push* for a stack implemented as a linked list?
O(1)

(i) Let the class Red be a subclass of the class Blue, and the class Green be a subclass of Red. True or false? The following statement is legal C++:

```
Green* green = new Blue();
```

False

(j) Again, let the class Red be a subclass of the class Blue, and the class Green be a subclass of Red. True or false? The following statement is legal C++:

```
Red* red = new Green();
```

True

Question 2 (10 points)

What are sentinels, what types of data structures are they used with, and how do they work?

They are extra (fake) list nodes placed at the two ends of a list. Used with Linked Lists and Doubly-Linked List. When a traversal reaches a sentinel, it knows it should stop. They also make inserts and deletes at the list ends uniform with other inserts and deletes (no special-case code for the end of the list).

Question 3 (20 points)

Consider the following C++ classes:

```
class Frog {
public:
    string call() {
        return "ribbit.";
    }
    virtual string alert() {
        return "breedeet!";
    }
};

class TreeFrog : public Frog {
public:
    string call() {
        return "eeeeep.";
    }
    string alert() {
        return "croak!";
    }
};
```

And the following code, inside main():

```
Frog* frog = new Frog();
Frog* frog2 = new TreeFrog();
TreeFrog* treeFrog = new TreeFrog();

cout << frog->call() << endl;
cout << frog->alert() << endl;
cout << frog2->call() << endl;
cout << frog2->alert() << endl;
cout << treeFrog->call() << endl;
cout << treeFrog->alert() << endl;
```

what is output to cout as a result of executing this code?

ribbit.
breedeet!
ribbit.
croak!
eeeeep.
croak!

Question 4 (20 points)

What is the (worst-case) time complexity of the pseudocode function *Multiply* below? Assume that all matrices (A, B, C, M1, and M2) are *n* by *n*. Express in O-notation. Show your work.

Let T(n) be the time for Multiply on n by n arrays.

Let U(n) be the time for findOneEntry on n by n arrays.

```
Multiply( A, B, C) { // A = B * C
        for(i = 1 to n) {
                for(j = 1 to n) {
                        A[i, j] = findOneEntry(B, C, i, j)
                }
        }
)

findOneEntry(M1, M2, I, j) {
        sum = 0;
        for(k = 1 to n) {
                sum += B[k, j] * C[i, k];
        }
        return sum
}
```

n iterations

n iterations

U(n) time

O(1)

n iterations

O(1)

O(1)

For Multiply: T(n) is n iterations of n iterations of U(n) work = $n^2U(n)$ work.

For findOneEntry: U(n) is O(1) + n iterations of O(1) work + O(1) = O(1) + O(n) + O(1) = O(n).

Since $T(n) = n^2U(n)$ and $U(n) \in O(n)$,

$$T(n) \in n^2O(n) = O(n^3)$$

Answer: $O(n^3)$

Question 5 (20 points)

What is the (worst-case) time complexity of the pseudocode function *pony* below? *n* is the size of the array A. Express in O-notation. Show your work.

```
pony(A) {
        horse(A, 0, n);
}
horse( A, i, j) {
        if(j – i  <= 1) {
                return;
        }

        Int k = (i + j) / 2;
        horse( A, i, k);
        horse( A, k, j);

        if(A[i] > A[k]) {
                swap(A, i, k);
        }
)
```

O(1)

O(1)

T(n/2)

T(n/2)

O(1)

Let T(n) be the time for horse on n = j-i+1 elements

So the time for horse, T(n), is:

$T(n) = O(1) + O(1) + T(n/2) + T(n/2) + O(1)$

$= 2T(n/2) + O(1)$

Use the Master Theorem: $k = \log_b a = \log_2 2 = 1$.

Is $O(1)$ in $O(n^{k-\varepsilon})$?

$O(n^{k-\varepsilon}) = O(n^{1-\varepsilon})$

So is $O(1)$ in $O(n^{1-\varepsilon})$? Yes, this is true for any $0 < \varepsilon < 1$.

This is therefore case 1 of the master theorem, and

$T(n) = O(n^k) = O(n)$

The time for pony is the time for horse on n-0 = n elements. Thus the time for pony is O(n).