COP 4530 Term Exam 2
Fall 2001
Professor Chris Lacher

Name: _____
SSN: _____
CS Username: _____
Score: _____ /100

This test contains 10 questions on 4 pages. Each question is worth 10 points.

1. In which of the following situations does a *constructor* get called? (Circle *all* correct answers.)

   A. When a function is called with a *value* parameter
   B. When a function is called with a *reference* parameter.
   C. When an object identifier comes *into scope*
   D. When an object identifier goes *out of scope*
   E. When an object is created dynamically using new
   F. When an object previously created by new is destroyed using delete
   G. When a function *returns a value*

2. In which of the following situations does a *copy constructor* get called? (Circle *all* correct answers.)

   A. When a function is called with a *value* parameter
   B. When a function is called with a *reference* parameter.
   C. When an object identifier comes *into scope*
   D. When an object identifier goes *out of scope*
   E. When an object is created dynamically using new
   F. When an object previously created by new is destroyed using delete
   G. When a function *returns a value*

3. In which of the following situations does a *destructor* get called? (Circle *all* correct answers.)

   A. When a function is called with a *value* parameter
   B. When a function is called with a *reference* parameter.
   C. When an object identifier comes *into scope*
   D. When an object identifier goes *out of scope*
   E. When an object is created dynamically using new
   F. When an object previously created by new is destroyed using delete
   G. When a function *returns a value*

4. For each of the following algorithms, give the answer that best describes the asymptotic runtime.

   a. _____

```
for (int i = 0; i < n; ++i)
  cout << i;
```

   b. _____

```
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
    cout << i << j;
```

The possible answers are:

A1:	$\Theta(1)$	A2:	$O(1)$
B1:	$\Theta(n)$	B2:	$O(n)$
C1:	$\Theta(n^2)$	C2:	$O(n^2)$
D1:	$\Theta(n^3)$	D2:	$O(n^3)$
E1:	$\Theta(\log n)$	E2:	$O(\log n)$
F1:	$\Theta(n \log n)$	F2:	$O(n \log n)$
G1:	$\Theta(\sqrt{n})$	G2:	$O(\sqrt{n})$

H: None of the above

c. _____

```
int i(n), j(0);
while (i > 1)
{
  i = i/2;
  ++j;
}
return j;
```

d. _____

```
bool hit (int n, int k)
for (i = 0; i < n; ++i)
{
  if (i == k) return true;
}
return false;
```

The possible answers are:

A1:	$\Theta(1)$	A2:	$O(1)$
B1:	$\Theta(n)$	B2:	$O(n)$
C1:	$\Theta(n^2)$	C2:	$O(n^2)$
D1:	$\Theta(n^3)$	D2:	$O(n^3)$
E1:	$\Theta(\log n)$	E2:	$O(\log n)$
F1:	$\Theta(n \log n)$	F2:	$O(n \log n)$
G1:	$\Theta(\sqrt{n})$	G2:	$O(\sqrt{n})$

H: None of the above

5. A Vector is illustrated in the first two rows below. Illustrate the binary search algorithm by filling in successive search ranges in the subsequent rows, with search value `ns`.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cb	dc	de	ha	hk	hm	hp	hx	jk	kk	ma	ns	oo	po	qo	sa

6. What is the return value of each of the following algorithms for the data of the previous question?

a. `lower_bound` _____
b. `upper_bound` _____
c. `binary_search` _____

For the next two questions, assume you are creating a *client* of `TList<>`.

7. Write a code fragment that declares a list `L` of `char` elements and inserts the letters `'a', 'c', 't'` (in that order) so that `L.Display()` results in sending `"cat"` to screen.

8. Complete the body of the following function that searches a list of String objects for a given object S

```
bool search (const TList<String>& L, const String& S)
{

}
```

For the next two questions, assume you are implementing the classes List<T> and ListIterator<T> using the definitions shown.

```
template < typename T >
class ListIterator
{
 friend class List <T>;
 private:
   List<T>::Link * current;
 public:
 // various public methods
};
```

```
template < typename T >
class List
{
 friend class ListIterator <T>;
 class Link
 {
  friend class List <T>;
  friend class ListIterator <T>;
  Link * prev, * next;
  T data;
  Link (const T& t)
   : data(t), prev(0), next(0){}
 };
 private:
  Link * first, * last;
 public:
 // various public methods
};
```

9. Supply the missing fragment of code in the implementation (ignoring possible failed memory requests):

```
template <typename T>
void List<T>::PushBack (const T& t)
{
  if (first == 0)
  {// this is the fragment to be supplied




    return;
  }
  // yada dada
```

10. Give the implementation of the following operator for `ListIterator<T>`:

```
template <typename T>
ListIterator<T>& ListIterator<T>::operator ++()
{




  return *this;
}
```