

Name: _____

Student ID number: _____

CS 14 – Midterm
100 points possible - 47 questions on 12 pages

For all questions, $\lg N$ or $\log N$ is the log base 2 of N .

For all questions, assume all ADT implementations and algorithms are good implementations.

Do NOT make any stray marks on your answer sheet. You may use pen or pencil, but if you want to change an answer, you MUST erase completely. Marks that are “X”ed out will be counted as filled. Your answer sheet will be scanned and graded automatically.

True/False questions. Please mark A for True and B for False on the answer sheet provided: (1 point each – 15 points total)

(For questions 1-40, plan to spend no more than 30 seconds per question on average = 20 minutes total)

For the following two questions, suppose $f(N) = O(h(N))$ and $g(N) = O(h(N))$.

1. Is $f(N) + g(N) = O(h(N))$?

2. Is $f(N) * g(N) = O(h(N))$?

3. If x and y are real numbers such that $0 < x < y$ then n^x is $O(n^y)$ but n^y is not $O(n^x)$

4. A stack is a FIFO structure.

5. A queue is a LIFO structure.

6. Assume you have a queue implemented with a singly linked list with head and tail pointers. Performing enqueue operations at the head and dequeue operations at the tail is more efficient than performing enqueue operations at the tail and dequeue operations at the head. (*Source – homework question*)

7. You can only write an iterator for a linked list. (*Source – textbook*)

8. Binary search has the same runtime on a sorted array as on a sorted linked list. (*Source – Study hint during lecture*)

9. $O(\max\{f(n), g(n)\}) = O(f(n) + g(n))$

10. n is $o(n \log n)$ (Notice that this question is asking about little-oh and NOT big-oh)

11. Anything that can be solved by using queues can be solved by using stacks.

12. Given the following class definition:

```
class DerivedClass : inheritanceType BaseClass
```

If you use *public inheritance*, public and protected members of the base class are private members of the derived class

13. Given the following class definition:

```
class DerivedClass : inheritanceType BaseClass
```

When using inheritance, the derived class constructor is the first constructor to be called.

14. Classes are a good example of polymorphism:

15. Operator overloading is a good example of encapsulation

Multiple choice questions. Please mark the answer on the answer sheet provided:
(2 pts each) – 50 points total

16. Put the following in order of **increasing** growth rate:

n. 2^n , $n \log n$, n^n , $n - n^3 + 7n^5$, $n^2 + \log n$, n^2 , $\log n$, $n!$

- A. n , $n^2 + \log n$, $\log n$, n^2 , $n \log n$, $n - n^3 + 7n^5$, $n!$, 2^n , n^n
- B. $n \log n$, n , $\log n$, n^2 , $n^2 + \log n$, n^2 , $n - n^3 + 7n^5$, 2^n , n^n , $n!$
- C. $n \log n$, $\log n$, n , $n - n^3 + 7n^5$, $n^2 + \log n$, n^2 , 2^n , $n!$, n^n
- D. 2^n , $n!$, n^n , $\log n$, n , $n \log n$, $n - n^3 + 7n^5$, $n^2 + \log n$, n^2
- E. $\log n$, n , $n \log n$, $n^2 + \log n$, n^2 , $n - n^3 + 7n^5$, 2^n , $n!$, n^n

17. An uncaught exception (one that matches no catch blocks) will (*Source – textbook and quiz 1*)

- A. End the function where it was thrown
- B. Terminate the program
- C. Continue execution
- D. Do nothing
- E. None of the above
- F. Not enough information

18. In C++, the protected modifier for members of a class C means:

- A. The exact same as private
- B. The same as private except when C inherits from another class
- C. The same as private except when another class derives from C
- D. There may only be one copy of the class in memory at a time
- E. All of the above
- F. None of the above

19. If the definition of class *Dog* contains the line “*friend class Cat*” then

- A. Instances of type *Dog* can access private members of instances of type *Cat*
- B. Instances of type *Cat* can access private members of instances of type *Dog*
- C. Instances of type *Dog* can access friend members of instances of type *Cat*
- D. Instances of type *Cat* can access friend members of instances of type *Dog*
- E. All of the above
- F. None of the above

20. Suppose you have a sorted array of 1000 elements. Using the most efficient search method possible, what is the maximum number of elements you will have to examine to determine if some value v is in the array?

- A. 1
- B. 9
- C. 10
- D. 250
- E. 500
- F. 1000

21. Lists may be either array-based or pointer-based (linked lists). Which of the following specify an **advantage** of a **pointer-based** implementation of a list over an array-based implementation of a list:

- A. Less space for each item
- B. Faster inserts and removes in the middle of the list
- C. No unused space
- D. A and B
- E. B and C

22. Lists may be either array-based or pointer-based (linked lists). Which of the following specify an **advantage** of an **array-based** implementation of a list over a pointer-based implementation of a list:

- A. Less space for each item
- B. Faster inserts and removes in the middle of the list
- C. No unused space
- D. A and B
- E. B and C

Consider the following class definitions:

```
class Sphere {  
public:  
    virtual double getArea ();  
    void displayStatistics ();  
};
```

```
class Ball: public Sphere {  
public:  
    double getArea ();  
    void displayStatistics ();  
};
```

Suppose that the implementation of each version of `displayStatistics` invokes the function `getArea`. Given the statements:

```
Ball myBall;  
SpherePtr = &myBall;
```

23. Which version of `displayStatistics` will the call `spherePtr->displayStatistics()` invoke?

- A) the Sphere class `displayStatistics` member function
- B) the Ball class `displayStatistics` member function

24. Which version of `getArea` will the call `spherePtr->displayStatistics()` invoke?

- A) the Sphere class `getArea` member function
 - B) the Ball class `getArea` member function
-

For the next 5 questions, use the following answer choices:

- | | | | |
|----|-------------|----|-------------------|
| A. | $O(1)$ | B. | $O(N)$ |
| C. | $O(N^2)$ | D. | $O(N \log N)$ |
| E. | $O(\log N)$ | F. | None of the above |

Give the Big-Oh runtime for the following snippets of code:

25. `void Ex2(int n)`
 `int a;`
 `for (int x = 0; x < n; x+=2) a = x;`
26. `void Ex3(int n)`
 `int a;`
 `for (int x = 0; x < n*n; ++x) a = x;`
27. `void Ex4(int n)`
 `int a;`
 `for (int x = 0; x < n; ++x)`
 `for (int y = 0; y < x; ++y) a = x;`
28. `void Ex5(int n)`
 `int a;`
 `for (int x = 0; x < n*n; ++x)`
 `for (int y = 0; y < x; ++y) a = x;`
29. `void Ex6(int n)`
 `int a;`
 `for (int x = 0; x < n; x++)`
 `for (int y = 1; y < n; y *= 2) a = x;`

 `for (int x = 0; x < 3*n; x++) a = x;`
-

For the next 6 questions, use the following answer choices:

- | | | | |
|----|-------------|----|-------------------|
| A. | $O(1)$ | B. | $O(N)$ |
| C. | $O(N^2)$ | D. | $O(N \log N)$ |
| E. | $O(\log N)$ | F. | None of the above |

30. Which of the above best represents the Big-Oh runtime for push into an array implemented stack?

31. Which of the above best represents the Big-Oh runtime for enqueue into a linked list implemented queue?

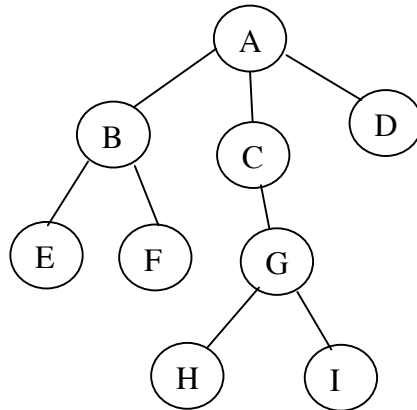
32. Which of the above best represents the Big-Oh runtime for search in an unsorted array?

33. Which of the above best represents the Big-Oh runtime for push back in a doubly linked list with only a head pointer (including search time)?

34. Which of the above best represents the Big-Oh runtime for `replaceAtRank` for an array implemented sequence?

35. Which of the above best represents the Big-Oh runtime for `replaceAtRank` for a linked list implemented sequence?

For the next 5 questions, use the following tree:



For all questions, height, depth, and path length are all edge based.

36. Which node is the root of the entire tree

- | | | | |
|----|------------------|----|-------------------|
| A. | A | B. | B |
| C. | E | D. | H |
| E. | All of the above | F. | None of the above |

37. Which node/nodes is/are descendant(s) of C (Mark only one answer)

- | | | | |
|----|------------------|----|-------------------|
| A. | A | B. | E |
| C. | G | D. | D |
| E. | All of the above | F. | None of the above |

38. Which node/nodes is/are ancestor(s) of C (Mark only one answer)

- | | | | |
|----|------------------|----|-------------------|
| A. | A | B. | E |
| C. | G | D. | D |
| E. | All of the above | F. | None of the above |

39. What is the path length of the path from A to I

- | | | | |
|----|---|----|-------------------|
| A. | 1 | B. | 2 |
| C. | 3 | D. | 4 |
| E. | 5 | F. | None of the above |

40. What is the height of node D

- | | | | |
|----|---|----|-------------------|
| A. | 1 | B. | 2 |
| C. | 3 | D. | 4 |
| E. | 5 | F. | None of the above |

(You should plan to spend no more than 5 minutes on this page)

Briefly define the following:

41. (3 pts) Object Oriented Programming

42. (3 pts) Tree

43. (3 pts) Static/early binding

44. (3 pts) Abstract Data Type

(You should plan to spend no more than 5 minutes on this page)

45. (9 pts) Given the following snippet of code:

```
int x = 1;  
int* ptr1 = NULL;  
int* ptr2 = 0;  
int* ptr3 = &x;  
int* ptr4 = x;  
*ptr3 = 2;
```

Describe what the following statements will produce (either what will be printed or what will happen).

a) `cout << ptr3 << endl;`

b) `cout << &x << endl;`

c) `cout << *ptr3 << endl;`

d) `cout << x << endl;`

e) `cout << ptr4 << endl;`

f) `cout << ptr1 << endl;`

g) `cout << ptr2 << endl;`

h) `cout << *ptr1 << endl;`

i) `cout << *ptr2 << endl;`

i) `cout << *ptr4 << endl;`

(You should plan to spend no more than 5 minutes on this page)

46. (6 pts) Write a recursive function to print the elements of a linked list **backwards** (print last element first and first element last). Be sure to use good programming style. You **must** use the function prototype provided and you may assume the following classes exist: **(you may not write pseudo-code)**

```
class List {
private:
    Node* head;
public:
    void printBackwards(Node*);
};

class Node {
public:
    itemtype item;
    Node* next;
};
```

```
void printBackwards ( Node* ptr ) {
```

(You should plan to spend no more than 8 minutes on this page)

47. (8 pts) Given an n-element unsorted linked list and a value k , write a *recursive* function to find and return a pointer to the node containing the **smallest value greater** than k . If no such value exists, return NULL. Your function may not use any global variables, that is, it can only use local variables and values passes as parameters. Use good programming style and write a general implementation. You must write code and not pseudo code. Syntax will be graded. (*source – slight modification to a question on quiz 1*)

You may assume that the following classes exist. The function you write will be a member function of the list class:

```
class List {
private:
    Node* head;
public:
    // your function
};

class Node {
public:
    Node* next;
    itemtype item;
};
```