

University of Nevada, Las Vegas Computer Science 477/677 Fall 2026

Study for Examination October 8, 2026

1. Fill in each blank. Write Ω if that is correct; otherwise write O or Ω , whichever is correct. Recall that $\log$ means $\log_2$.

- (a) $n^{.01} = \text{-----}(\log n)$
- (b) $n^{\log n} = \text{-----}(2^{(\log n)^2})$
- (c) $\sum_{i=1}^n i^k = \text{-----}(n^{k+1})$
- (d) $\log n^2 = \text{-----}(\log n^3)$
- (e) $\log(n!) = \text{-----}(n \log n)$
- (f) $\log n = \text{-----}(\ln n)$
- (g) $\log n = \text{-----}(\log \log n)$

2. Fill in the blanks.

- (i) No good programmer would ever use an unsorted list as a search structure. ----- (True or False.)
- (ii) Heapsort is a fast version of -----
- (iii) Treesort is a fast version of -----
- (iv) Shell sort is inspired by ----- but is faster.
- (v) For any sorting algorithm where all branch points in the code are comparisons, the number of comparisons while sorting n items is ----- (Use Ω notation.)
- (vi) Name two different divide-and-conquer sorting algorithms. -----
- (vii) Name three kinds of priority queues: -----
- (viii) The worst case asymptotic time complexity of sorting an array of length n is:
 - i. ----- using bubblesort.
 - ii. ----- using quicksort.
 - iii. ----- using heapsort.
- (ix) The asymptotic time complexity to find an item in an ordered array of length n is ----- using the divide and conquer algorithm -----
- (x) In a priority queue, each item in the structure represents an -----
- (xi) The height of a binary tree with 20 leaves must be at least ----- . Exact answer please: no partial credit.

3. There are three ways that we've discussed in class to handle the *false overflow* problem for a queue. What are they?

4. The following C++ code is for an array implementation of queue. Fill in the missing code for enqueue, dequeue, and empty.

```
struct queue
{
    int A[N]; // N is a constant large enough to prevent false overflow.
    int rear = 0;
    int front = 0;
};

void enqueue(queue&q,int newitem) // inserts newitem into q
{

}

bool empty(queue&q) // returns true if q is empty, false otherwise
{

}

int dequeue(queue&q) // returns an item from q and deletes that item
{

}
```

5. What follows is part of a C++ implementation of binary tree of integer. (It runs, I checked.) Fill in the missing code for the recursive functions `insert`, `find`, and `lcation`. The function `lcation` should be called `find`, but that would cause a redundancy conflict.

```
% First bst
struct treenode;
typedef treenode*tree;
struct treenode
{
    int item;
    tree lft; // left subtree
    tree rgt; // right subtree
};

void insert(tree&root,int n)
    // inserts n into the binary search tree. Duplicates are not inserted.
{
```

```

}

bool find(tree root, int n)
    // returns true if some node contains n, false otherwise.
{

}

tree lcation(tree root, int n)
    // returns pointer to the node which contains n, if any; nulptr otherwise.
{

}

int higt(tree root)
    // return the height of the binary search tree. If empty returns -1.
{

}

int kount(tree root)
    // return the number of item in the binary search tree.
{

}

void preordertraverse(tree root)
    // writes the items of tree in preorder.
{

}

void inordertraverse(tree root)
    // writes the items of tree in inorder.
{

}

void postordertraverse(tree root)
    // writes the items of tree in postorder.
{

}

```

6. A max-heap is implemented as a binary tree which is, in turn, is implemented as an array, which is shown in the first row of the table below. We execute two operations. Show the evolution of the heap when “Z” is inserted. Then, show the evolution of the heap when *deletemax* is executed.

8	W	H	R	E	G	F	Q	A	L	B

7. Rewrite the prefix expression $*a - b \sim c$ in infix notation and postfix notation.
8. Assume that the array `int A[N]` is part of the input. What does `sorted` do? What does `gcd` do? What does `ternary` do? What does `power` do?

```
bool sorted()
{
    bool rslt = true;
    for(int i = 1; i < N and rslt; i++)
        if(A[i] < A[i-1]) rslt = false;
    return rslt;
}

int gcd(int a, int b)
{
    //cout << "Computing gcd(" << a << ", " << b << ")" << endl;
    if(a < 0) return gcd(-a, b);
    else if(b < 0) return gcd(a, -b);
    else if(b < a) return gcd(b, a);
    else if(a == 0) return b;
    else return gcd(b%a, a);
}

void mainwork(int n)
{
    assert(n >= 0);
    if(n > 2) mainwork(n/3);
    cout << n%3;
}

void ternary(int n)
{
```

```

    mainwork(n);
    cout << endl;
}

float power(float x, int n)
{
    assert(n >= 0);
    float y = x;
    float z = 1.0;
    int m = n;
    while(m > 0)
    {
        if(m%2) z = z*y;
        y = y*y;
        m = m/2;
    }
    return z;
}

```

9. Show a circular queue with dummy node, with items A, B, C, D, in that order from front to rear. Then show how the queue changes when you insert E, and then show how the queue changes when you execute dequeue.
10. Find the constant C such that the n^{th} Fibonacci number is $\Theta(C^n)$.
11. The answer to each of these questions is either **bubbleup** or **bubbledown**. If a max-heap is implemented a binary tree implemented as an array, _____ is needed for insertion, while _____ is needed for deletemax.
12. For each of the code fragments below, find the asymptotic time complexity in terms of n , using Θ notation.
 - (a) `for(int i = 0; i*i < n; i++)`

(b) `for(float x = n; x > 2.0; x = sqrt(x))`

(c) `for(int i = 1; i < n; i=2*i)`

(d) `for(int i = n; i > 1; i=i/2)`

(e) `for(int i = n; i > 1; i--)`
 `for(int j = i; j > 1; j = j/2)`

(f) `for(int i = n; i > 1; i--)`
 `for(int j = n; j > i; j = j/2)`

(g) `for(int i = 2; i < n; i=i*i)`

(h) `for(int i = 0; i < n; i++)`
 `for(int j = 0; j < i; j++)`
 `cout << "Hello world" << end;`

(i) `for(int i = 0; i < n; i++)`
 `for(int j = 1; j < i; j = 2*j)`
 `cout << "Hello world" << end;`

(j) `for(int i = 0; i < n; i++)`
 `for(int j = i; j < n; j = 2*j)`
 `cout << "Hello world" << end;`

(k) `for(int i = 0; i < n; i++)`
 `for(int j = 0; j*j < i; j++)`
 `cout << "Hello world" << end;`

13. Solve each of these recurrences, giving the answer in terms of n using Θ notation.

(a) $F(n) = 2F(n/2) + n$

(b) $F(n) = 2F(n/4) + 1$

(c) $F(n) = 2F(n/2) + n^2$

(d) $F(n) = F(n - \log n) + \log n$

(e) $F(n) = F(n - \sqrt{n}) + n$

(f) $F(n) = F(\sqrt{n}) + 1$

14. Simplify each of these expressions. Hint: Answers are integers or simple fractions.

(a) $\log 0.125$

- (b) $\frac{\log_3 5 \log 3}{\log 5}$
- (c) $\log_4 8$
- (d) $\frac{4^{\log 3}}{9}$
- (e) $\log \log 16$
- (f) $2^{\log_4 25}$

15. The following code computes the product of its two parameters. Write a loop invariant of the loop.

```
int product(float x, int n)
{
    // input condition: n is non-negative
    float y = x;
    float z = 0.0;
    int m = n;
    while(m > 0)
    {
        if(m%2) z = z+y;
        y = 2*y;
        m = m/2;
    }
    return z;
}
```

16. Understand **Selection sort**, **Insertion sort**, **Bubble sort**, and **Shell sort**.

17. Walk through the steps of heapsort, sorting an array whose items are X,B,L,F,U,N in that order. For ease of grading, use the matrix below. The number of rows in the matrix might not be exactly what you need. If not, leave rows blank or add extra rows.

18. The following is an unfinished array implementation of stack of integer. Fill in the missing code for empty, push, and pop.

```
struct stak
{
    int item[N];
    int size;
};

void init(stak&S)
{
    S.size = 0;
}

bool empty(stak S)
{
}
```

```

void push(stak&S, int newitem)
{

}

int pop(stak&S)
{
    assert(not empty(S));

}

```

What do you assert in the code for pop?

19. The following is an unfinished linked list implementation of stack of integer. Finish it.

```

struct stacknode;
typedef *stacknode stak;
struct stacknode
{
    int item; Error Fixed Here
    stak link;
};
void init(stak&S)
{

}
bool empty(stak S)
{

}
void push(stak&S, int newitem)
{

}
int pop(stak&S)
{

}

```

20. Find an optimal binary prefix-free code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

A	8
B	5
C	3
D	11
E	14
F	2

21. Write pseudocode for the Bellman Ford algorithm. Be sure to include the shortcut.

22. Write pseudocode for the Floyd Warshall algorithm.