

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

Answers to Examination October 8, 2026

Name: \_\_\_\_\_

No books, notes, scratch paper, or calculators. Use pen or pencil, any color. Use the rest of this page and the backs of the pages for scratch paper. If you need more scratch paper, it will be provided. If you want anything on extra pages to be graded, staple those pages to your test and write, "Please grade this page."

The entire examination is 310 points.

1. True (T)/False (F)/Open(O) meaning that the answer is not known to science at this time.
  - (i) [5 points]  $\log n^2 = \Theta(\log n^3)$
  - (ii) [5 points]  $\log^2 n = O(\log^3 n)$
  - (iii) [5 points]  $\log n! = \Theta(n \log n)$
2. Fill in the blanks.
  - (i) [10 points] Two divide and conquer sorting algorithms we've studied this semester are **quicksort** and **mergesort**.
  - (ii) [5 points] Any comparison-based sorting algorithm on a file of size  $n$  executes  $\Omega(n \log n)$  comparisons. (Use  $\Omega$  notation.)
  - (iii) [15 points] The asymptotic time complexity to find an item in an ordered array of length  $n$  is  $O(n)$  using the divide and conquer algorithm **binary search**.
  - (iv) [10 points] In a priority queue, each item in the structure represents an **unfulfilled obligation**.
  - (v) [5 points] A binary tree of height 3 can have no more than 15 nodes. Exact answer please: no partial credit.
  - (vi) [5 points] In any shortest (minimum weight) path problem in a directed graph, it is impossible to find a solution if the graph contains a **negative cycle**.
  - (vii) [5 points] **Shellsort** uses a sequence of gaps; each phase uses one of those gaps to partition the initial array into arithmetic sequences, each of which is sorted using bubblesort.
  - (viii) [5 points]  $O(n^2)$  is the worst case asymptotic time complexity of quicksort on an array of size  $n$ . (Use  $O$  notation.)
  - (ix) [5 points] The asymptotic time complexity of merging two length  $n$  sorted arrays is  $O(n)$ . (Use  $O$  notation.)

3. [15 points] What follows is part of a C++ implementation of binary search tree, where the items are integers, and a tree is not allowed to contain duplicate items. Fill in the missing code for the indicated functions.

```
struct bstnode;
typedef bstnode*bst;
struct bstnode
{
    int item;
    bst left;
    bst right;
};

void insert(bst&root, int n)
// inserts the item n into the binary search tree rooted at root, provided
// that item is not already in the tree.
{
    if(!root)
    {
        root = new(bst);
        root->item = n;
    }
    else if(n < root->item) insert(root->left,n);
    else if(n > root->item) insert(root->right,n);
}

bool find(bst root, int n)
// returns true if the item of some node is n, false otherwise.
{
    if(root)
        if(root->item == n) return true;
        else if(n < root->item) return find(root->left,n);
        else return find(root->right,n);
    else return false;
}

\eject
void postordertraverse(bst root)
//Writes the items of the tree rooted at root in postorder.
{
    if(root)
    {
        postordertraverse(root->left);
        postordertraverse(root->right);
    }
}
```

```

        cout(root->item) << " ";
    }
}

```

4. [20 points] A maxheap is implemented as an almost complete binary tree which is implemented as an array. The data structure also contains a number indicated the current number of items in the heap, which can be larger than the array. Show the evolution of the heap shown below when “Y” is inserted. Then, show the evolution of the heap when *deletemax* is executed.

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
| 6 | X | V | T | Q | H | K | R | D | S | Z |
| 7 | X | V | T | Q | H | K | Y | D | S | Z |
| 7 | X | V | Y | Q | H | K | T | D | S | Z |
| 7 | Y | V | X | Q | H | K | T | D | S | Z |
| 6 | T | V | X | Q | H | K | T | D | S | Z |
| 6 | X | V | T | Q | H | K | T | D | S | Z |
|   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |

5. [10 points] The following is C++ code for what well-known algorithm?  
Selection sort.

```

void mystery1()
{
    for(int i = 0; i < N-1; i++)
        int(for j = i+1; j < N; j++)
            if(A[j] < A[i]) swap(A[i],A[j]);
}

```

6. [10 points] The following C++ computes a well-known function. Name that function.

```

int mystery2(int n)
{
    assert(n > 0);
    int rslt = 0;
    int m = n;
    while(m > 1)
    {
        m = m/2;
        rslt++;
    }
    return rslt;
}

```

Base 2 logarithm.

7. [10 points] What is the output of the following C++ function?

```
void mystery3(int n)
{
    assert(n >= 0)
    if(n > 1) mystery3(n/2);
    cout << n%2;
}
```

The binary numeral for  $n$ .

8. Consider the following C++ code.

```
float mystery4(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;
}
```

- (i) [10 points] What is the output of the function `mystery4`?

$$x^n$$

- (ii) [10 points] What is the loop invariant of the loop?

$$x^n = y^m * z$$

9. [20 points] Write pseudocode for the Bellman Ford algorithm. Be sure to include the shortcut.

Let  $W[u,v]$  be the weight of the arc from  $u$  to  $v$ .  $V[i]$  be the least weight of any path from 0 to  $i$ .

$V[0] = 0$

for all vertices  $i$

$V[i] = \infty$

$V[0] = 0$

finished = false

while(not finished)

    finished = true

        for all arcs  $[u,v]$

            temp =  $V[u] + W[u,v]$

            if(temp <  $V[v]$ )

$V[v] = \text{temp}$

                back[v] =  $u$

                finished = false

10. Write pseudocode for the Floyd Warshall Algorithm.

Let  $W[i,j]$  be the weight of the arc from  $i$  to  $j$ , if no such arc,  $\infty$ .

Let  $V[i,j]$  be the weight of the minimum weight path from  $i$  to  $j$  found so far, and let back[i,j] be the next-to-the-last vertex on the that path.

for all  $i$

    for all  $j \neq i$

$V[i,j] = W[i,j]$

        back[i,j] =  $i$

for all  $j$

    for all  $i$

        for all  $k$

            temp =  $V[i,j] + V[j,k]$

            if(temp <  $V[i,k]$ )

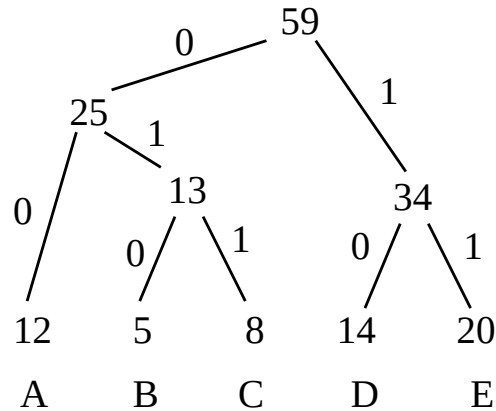
$V[i,k] = \text{temp}$

                back[i,k] = back[j,k]

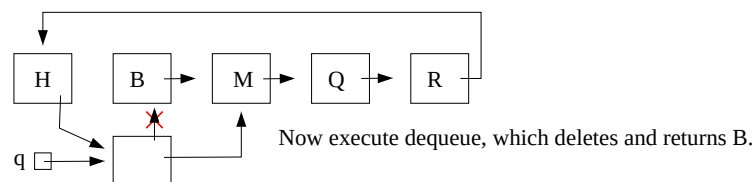
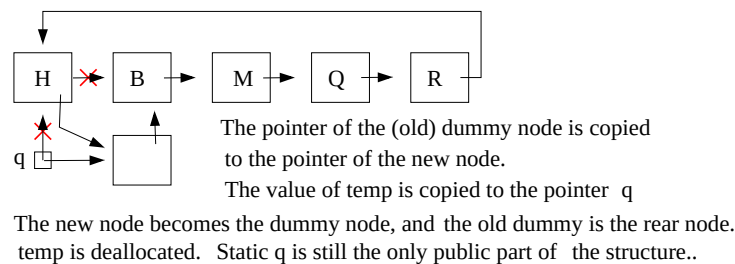
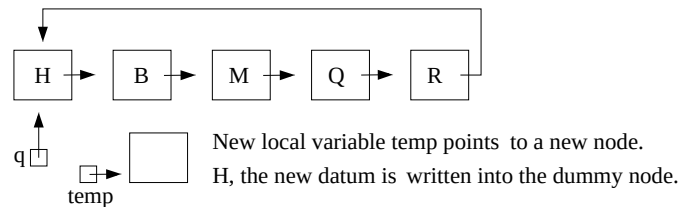
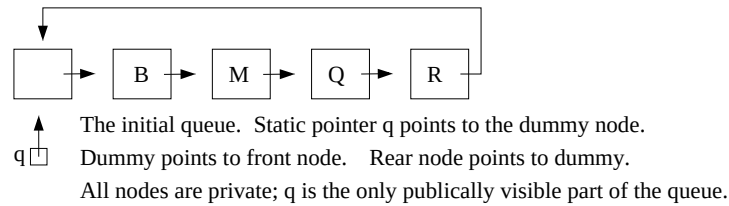
11. [20 points] Compute an optimal prefix-free binary code for the alphabet  $\{A, B, C, D, E\}$  with frequencies shown in the table below. Then use that code to encode the message “ACEDEBD.”

0001111100001010

|   |    |     |
|---|----|-----|
| A | 12 | 00  |
| B | 5  | 010 |
| C | 8  | 011 |
| D | 14 | 10  |
| E | 20 | 11  |



12. [20 points] Show a circular queue with dummy node, with items B, M, Q, R, in that order from front to rear. Then show how the queue changes when you insert H, and then show how the queue changes when you execute dequeue.



13. [10 points] The answer to each of these questions is either **bubbleup** or **bubbledown**. If a max-heap is implemented a binary tree. **bubbledown** is needed for insertion, while **bubbleup** is needed for deletemax.

14. For each of the code fragments below, find the asymptotic time complexity in terms of  $n$ .

(i) [5 points]

```
for(float x = n; x > 1; x = x/2)
```

$\Theta(\log n)$

(ii) [5 points]

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

$\Theta(\sqrt{n})$

(iii) [5 points]

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

$\Theta(\log \log n)$

(iv) [5 points]

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

$\Theta(\log n)$

(v) [5 points]

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

$\Theta(\log n)$

(vi) [5 points]

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

$\Theta(n \log n)$

(vii) [5 points]

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

$\Theta(n)$

(viii) [5 points]

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

$\Theta(\log \log n)$

15. Give asymptotic solutions to the following recurrences.

(i) [5 points]  $F(n) = 2F(n/2) + n$

$$F(n) = \Theta(n \log n)$$

(ii) [5 points]  $F(n) = 2F(n/2) + 1$

$$F(n) = \Theta(n)$$

(iii) [5 points]  $F(n) = 2F(n/2) + n^2$

$$F(n) = \Theta(n^2)$$

(iv) [5 points]  $F(n) = 4F(n/2) + n^2$

$$F(n) = \Theta(n^2)$$

(v) [5 points]  $F(n) = F(\sqrt{n}) + 1$  Hint: use substitution.

Let  $m = \log n$ , and  $F(n) = G(m) = G(\log n)$ . Then  $F(\sqrt{n}) = G(\log \sqrt{n}) = G((\log n)/2) = G(m/2)$

We get the recurrence:

$$G(m) = G(m/2) + 1. \text{ Finally}$$

$$F(n) = G(m) = \Theta(\log m) = \Theta(\log \log n)$$

(vi) [5 points]  $F(n) = F(n-1) + n^2$

Subtract  $F(n-1)$  and divide both sides by 1.

$$\frac{F(n) - F(n-1)}{1} = n^2$$

$$F'(n) = n^2 \text{ (asymptotically, of course)}$$

$$F(n) = \Theta(n^3)$$

(vii) [5 points]  $F(n) = 3F(n/9) + 1$

$$F(n) = \Theta(\sqrt{n})$$