University of Nevada, Las Vegas Computer Science 477/677 Fall 2025

Answers to Examination February 12, 2025

The entire examination is 260 points.

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) [5 points] $\log(2n) = \Theta(\log(3n))$

   (b) [5 points] $10 \log n = \Theta(\log(n^2))$

   (c) [5 points] $n^{.01} = \Omega(\log n)$

   (d) [5 points] $n^{\log n} = \Theta(2^{(\log n)^2})$

   (e) [5 points] $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$

   (f) [5 points] $\log n^2 = \Theta(\log n^3)$

   (g) [5 points] $\log(n!) = \Theta(n \log n)$

   (h) [5 points] $\log n = \Theta(\ln n)$

   (i) [5 points] $\log n = \Omega(\log \log n)$

2. Fill in the blanks.

   (i) [5 points] Any comparison-based sorting algorithm on a file of size n executes $\Omega(n \log n)$ comparisons. (Use Ω notation.)

   (ii) [5 points] The asymptotic time complexity to find an item in an ordered array of length n is $O(\log n)$ using the divide and conquer algorithm **binary search**.

   (iii) [5 points] In a priority queue, each item in the structure represents an **unfulfilled obligation**.

   (iv) [5 points] The height of a binary tree with 9 leaves must be at least **4**. Exact answer please: no partial credit.

3. [5 points] There are three ways that we've discussed in class to handle the *false overflow* problem for a queue. What are they?

   wrap slide bigger

4. [15 points] The following portion of C++ code contains an array implementation of `queue`. Fill in missing code for the operators "`enqueue`," '`dequeue`" and "`empty`."

```
struct queue
 {
  int A[N]; // N is a constant large enough to prevent false overflow.
  int rear = 0; // initially the queue is empty
  int front = 0; // position of the front item
```

```
 };

void enqueue(queue&q,int newitem) // inserts newitem into q
 {
  q.A[q.rear] = newitem;
  q.rear++;
 }

bool empty(queue&q) // returns true if q is empty, false otherwise
 {
  return q.rear == q.front;
 }

int dequeue(queue&q) // returns an item from q and deletes that item
 {
  assert(not empty(q));
  // alternatively, assert q.front < q.rear;
  int rslt = q.A[q.front];
  q.front++;
  return rslt;
 }
```

5. [15 points] 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.

```
struct treenode;
typedef treenode*tree;
struct treenode
 {
  int item;
  tree left;
  tree right;
 };

tree mainroot; // the root of the binary search tree

void insert(tree&root,int n)
  // inserts n into the bineary search tree.  Duplicates are not inserted.
 {
  if(root)
   if(n < root->item) insert(root->left,n);
   else if(n > root->item) insert(root->right,n);
```

```
   else;
  else
   {
    root = new rootnode;
    root->item = n;
   }
 }

bool find(tree root, int n)
  // returns true if some node contains 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;
 }

tree lcation(tree root,int n)
  // returns pointer to node containng n, if any, otherwise NULL}
 {
  if(root)
   if(root->item == n)
    return root;
   else if(n < root->item) return lcation(root->left,n);
   else return lcation(root->right,n);
  else return NULL;
 }
```

6. [10 points] 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. In the subsequent rows, show the evolution of the heap when "Z" is inserted.

9	A	H	C	M	N	F	G	P	R	K
10	A	H	C	M	N	F	G	P	R	Z

7. [5 points] Rewrite the infix expression $-a * b - (-c * d + e) * f$ in prefix notation and postfix notation.

   $-* \sim ab * +* \sim cdef$ $\qquad\qquad$ $a \sim b * c \sim d * e + f * -$

8. [15 points] What well-known algorithm does the procedure `mystery1` implement? **selectionsort**

   What well-known algorithm does `mystery2` implement? **bubblesort**

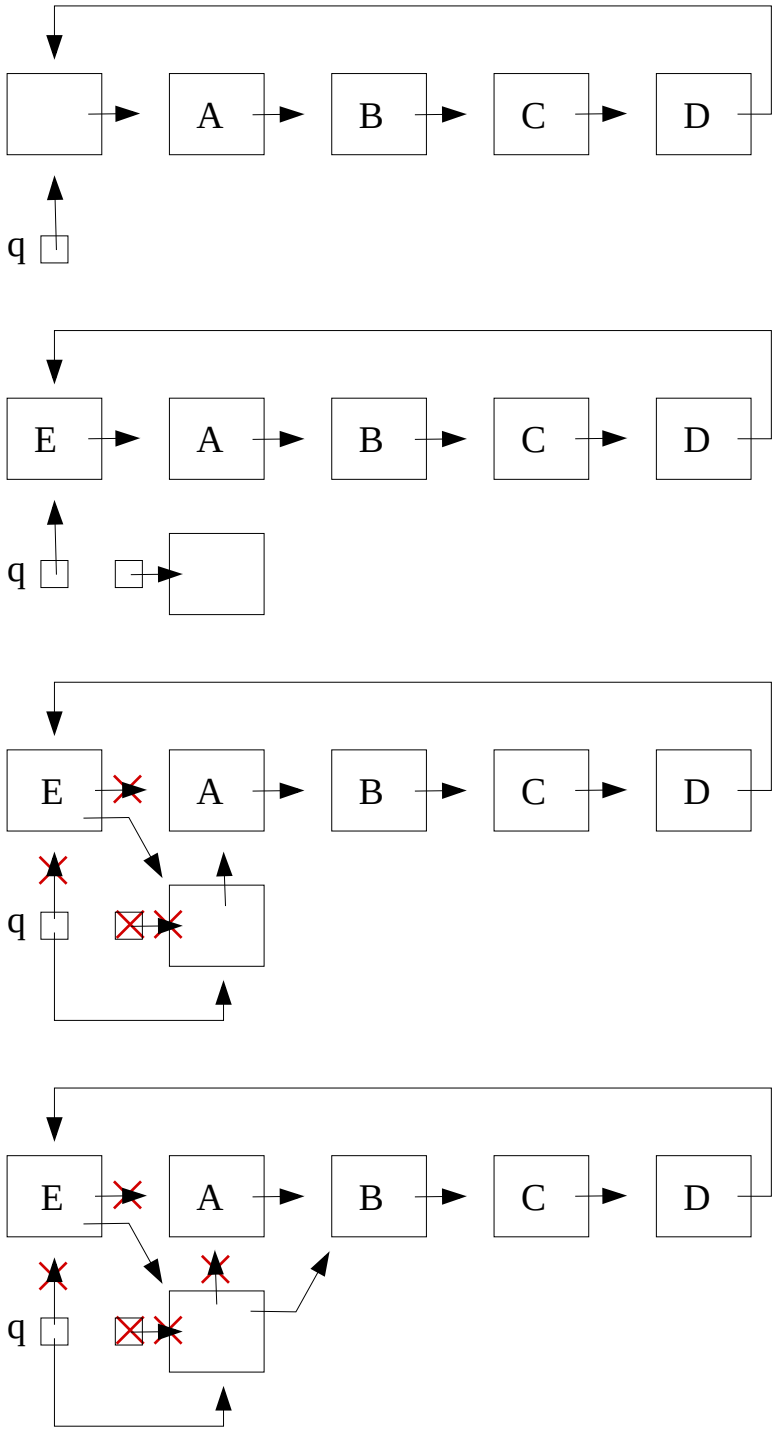
   What well-known algorithm does `mystery3` implement? **insertion sort**

   Assume that int A[N] has been declared.

```
void swap(int x, int y)
 {
  int z = x;
  x = y;
  y = z;
 }

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

void mystery2()
 {
  bool finshd = false;
  while(not finshd)
   {
    finshd = true;
    for(int i = 0; i < N-1; i++)
     if(A[i+1] < A[i])
      {
       finshd = false;
       swap(A[i],A[i+1]);
      }
   }
 }

void mystery3()
 {
  for(int i = 1; i < N; i++)
   {
    bool fnshd = false;
    for(int j = i; j > 0 and not finshd; j--)
     if(A[j] < A[j-1]) swap(A[j],A[j-1]);
     else fnshd = true;
```

```
  }
}
```

9. [10 points] 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. [10 points] Let $F_1, F_2, F_3 \ldots$ be the Fibonacci numbers Recall that $F_i + F_{i+1} = F_{i+2}$. The first few Fibonacci numbers are 1, 1, 2, 3, 5, 8, ...

    Find the smallest constant C such that $F_n = O(C^n)$.

    We can assume that $F_n = C^n$. Thus

$$\begin{aligned} F_n + F_{n+1} &= F_{n+2} \\ C^n + C^{n+1} &= C^{n+2} \\ 1 + C &= C^2 \\ C^2 - C - 1 &= 0 \\ C &= \frac{1 \pm \sqrt{1+4}}{2} \text{ by the quadratic formula} \\ &= \frac{1 + \sqrt{1+4}}{2} \text{ since we know } C \text{ must be positive} \end{aligned}$$

11. [10 points] The following C++ code implements a function.

```
int mystery4(int n)
 {
  int m = n;
  int rslt = 0;
  while(m > 0)
   {
    rslt = rslt + n;
    m--;
   }
  return rslt;
 }
```

    What is the value of mystery4(n)? What is the loop invariant of the loop?

    mystery4(n) = n^2, and the loop invariant is $n^2 = m * n +$ rslt.

12. The answer to each of these questions is either **bubbleup** or **bubbledown**.

    A max-heap is implemented a binary tree. **bubbleup** is needed for insertion, while **bubbledown** is needed for deletemax. The answers are the same for a min-heap.

13. [10 points] The following is C++ code for a function that computes a floating point number to the power of a positive integer. Find a loop invariant which can be used to prove correctness of the function.

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

The loop invariant is $x^n = y^m * z$

14. For each of the code fragments below, find the asymptotic time complexity in terms of n. Assume that the value of n is given.

(a) [5 points]

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

$\Theta(\log n)$

(b) [5 points]

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

$\Theta(\sqrt{x})$

(c) [10 points] `for(float x = n; x > 2.0; x = sqrt(x))`

$\Theta(\log \log n)$

(d) [5 points]

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

$\Theta(\log n)$

(e) [5 points]

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

$\Theta(\log n)$

(f) [10 points]

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

$\Theta(n \log n)$

(g) [10 points]

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

$\Theta(n)$

(h) [10 points]

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

$\Theta(n \log n)$

(i) [10 points]

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

$\Theta(n)$

(j) [10 points]

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

$\Theta(\log \log n)$

15. [10 points] Walk through polyphase mergesort with the following input:

I have inserted commas to separate the runs of a sequence.

D,BEFMPX,TY,W,N,LZ,G

DTY,N,G
BEFMPX,W,LZ

BDEFMPTXY,GLZ
NW

BDEFMNPTWXY
GLZ

BDEFGLMNPTWXYZ