University of Nevada, Las Vegas
Computer Science 477/677 Fall 2022
Answers to Final Examination December 15, 2022

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.

The entire test is 480 points.

1. Fill in the blanks.

   (i) [10 points] The following is pseudo-code for what algorithm?

   selection sort

```
int x[n];
obtain values of x;
 for(int i = n-1; i > 0; i++)
  for(int j = 0; j < i; j++)
   if(x[i] < x[j]) swap(x[i],x[j]);
```

   (ii) [10 points] **Dijkstra's** algorithm does not allow the weight of any arc to be negative.

   (iii) [10 points] The time complexity of every comparison-based sorting algorithm is $\Omega(n \log n)$. (Your answer should use Ω notation.)

   (iv) [10 points] The prefix expression $*a+ \sim b * -c\, d \sim e$ is equivalent to the infix expression $a * (- + (c - d) * -e)$ and the postfix expression $ab \sim cd - e \sim * + *$

   (v) [10 points] The items stored in a priority queue represent **unfulfilled obligations.**

   (vi) [10 points] The asymptotic complexity of Dijkstra's algorithm algorithm is $O(m \log n)$ or $O(m \log m)$ (Either answer is correct, but $O(n \log n)$ is wrong. Also, use of Θ or Ω is wrong.)

   (vii) [10 points] A **perfect** hash function fills the hash table exactly with no collisions.

   (viii) [10 points] In **open hashing** there can be any number of items at a given index of the hash table.

   (ix) [10 points] If the position at $h(x)$ is already occupied for some data item x, a **probe sequence** is used to find an unoccupied position in the hash table.

   (x) [10 points] In **cuckoo** hashing, each item has more than one hash value, but only uses one of them.

   (xi) [10 points] If G is a weighted directed graph, then it is impossible to solve the all pairs shortest path problem for G if G has a **negative weight cycle**. (Just **negative cycle** is sufficient.)

   (xii) [10 points] A planar graph with n vertices can have no more than $3n - 6$ edges if $n \geq 3$. (Exact formula, please.)

2. Give the asymptotic complexity, in terms of `n`, of each of the following code fragments. [10 points each]

(a)
```
for(int i = 2; i < n; i = i*i)
  cout << "Hello world" << endl;
```

$\Theta(\log \log n)$

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

$\Theta(n \log n)$

(c)
```
for(int i = 1; i*i < n; i++)
  cout << "hello world" << endl;
```

$\Theta(\sqrt{n})$

(d)
```
for(int i = 0; i < n; i++)
  for(int j = n; j > i; j = j/2)
   cout << "Hello world!" << endl;
```

$\Theta(n)$

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

$\Theta(n \log \log n)$

3. Solve the recurrences. Give the asymptotic value of $F(n)$ in terms of n, using Θ notation. [10 points each]

(a) $F(n) = 4F\left(\frac{n}{2}\right) + n$

$\Theta(n^2)$

(b) $F(n) \geq F\left(\frac{n}{2}\right) + 2F\left(\frac{n}{4}\right) + n$

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

(c) $F(n) \leq 2F(n/2) + n^2$

$F(n) = O(n^2)$

(d) $F(n) \geq 3F(n/9) + 1$

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

(e) $F(n) = F(3n/5) + 4F(2n/5) + n^2$

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

(f) $F(n) \leq 4F(n/2) + n^2$

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

(g) $F(n) \leq F(\sqrt{n}) + 1$

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

4. [20 points] List properties of a good hash function used for a search structure.

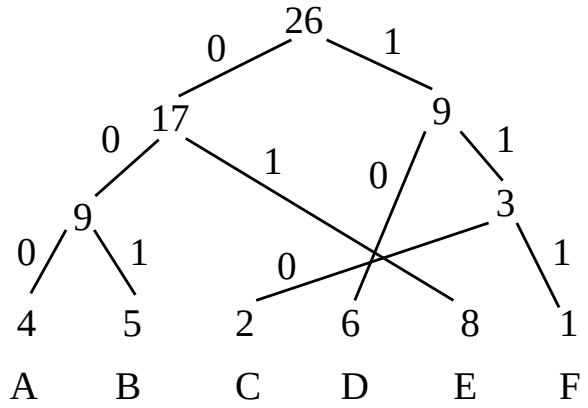
   (a) Deterministic,

   (b) Appears randomized, or irrelevant to the actual data,

   (c) Fast to compute.

5. [20 points] Find an optimal prefix code for the alphabet $\{A, B, C, D, E, F\}$ where the frequencies are given in the following array.

A	4	000
B	5	001
C	2	110
D	6	10
E	8	01
F	1	111



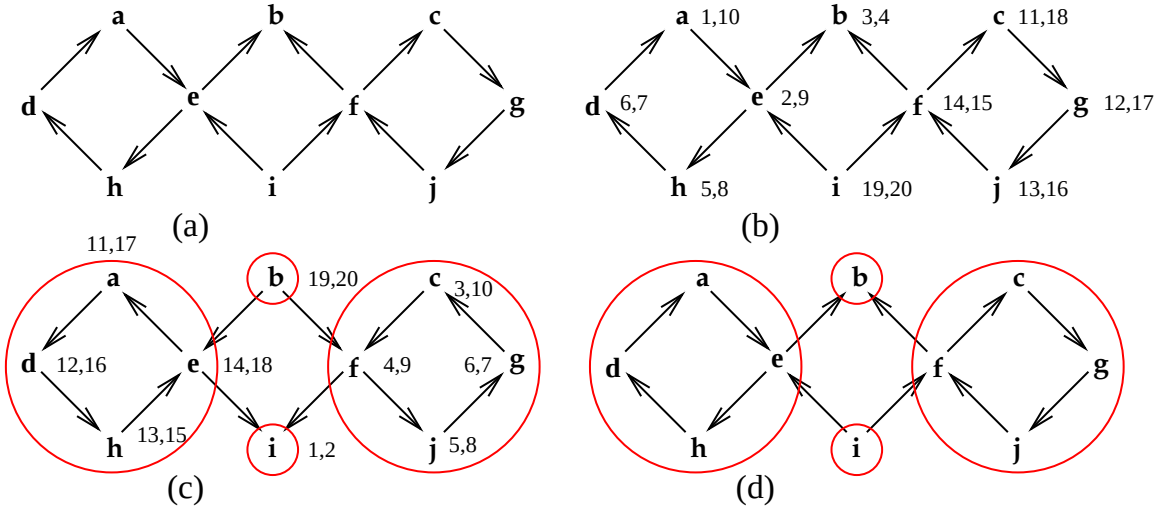
6. [20 points] Write pseudocode for the Floyd/Warshall algorithm.

Let the vertices be $1, 2, \ldots n$. Write $W[i, j]$ for the weight of the edge from i to j, which is ∞ if there is no such edge. The algorithm computes $V[i, j]$, the least weight of any path from i to j, and `back`$[i, j]$, the backpointer of a least weight path from i to j.

```
for all i and j
  {
   V[i,j] = W[i,j]
   back[i,j] = i
  }
for all i
  V[i,i] = 0
for all j
  for all i
    for all k
     {
      temp = V[i,j] + V[j,k]
      if(temp < V[i,k])
```

```
        {
         V[i,k] = temp
         back[i,k] = back[j,k]
        }
    }
```

7. [20 points] Use the DFS method to find the strong components of the digraph in the first figure below. Show steps, using the second figure as needed.



8. [20 points] What is the loop invariant of the loop in the following function?

$x * n = y * m + z$

```
float product(float x, int n)
 {
  float z = 0.0;
  float y = x;
  int m = n;
  while(m > 0)
   {
    if(m%2) z = z+y;
    m = m/2;
    y = y+y;
   }
  return z;
 }
```

9. [20 points] Write pseudocode for the Bellman-Ford algorithm. Be sure to include the shortcut that ends the program when the final values have been found.

   Assume there are m arcs, numbered from 0 to $m-1$, and the k^{th} arc is from $s[k]$ to $t[k]$ and has weight $w[k]$. The vertices are numbers $0 \ldots n-1$ and the source vertex is 0. For each vertex i, The algorithm computes $V[i]$, the minimum weight of a path from 0 to i, and the backpointer $b[i]$ of that path. The algorithm uses a shortcut.

```
V[0] = 0;
for(int i = 1; i < n; i++)
  V[i] = infinity;
bool done = false;
while (not done)
  {
   done = true;
   for(int k = 1
   for(int k = 0; k < m; k++)
    {
     temp = V[s[k]] + w[k];
     if(temp < V[t[k]])
      {
       V[t[k]] = temp;
       done = false;
      }
    }
  }
```

10. Consider the following C++ code.

```
int george(int n)
 {
  if(n == 0) return 1;
  else return george(n/2)+george(n/2-1)+n*n;
 }
```

   (a) [10 points] What is the asymptotic complexity of `george(n)`?

   $G(n) = G(n/2) + G(n/2) + n^2$. Thus $G(n) = \Theta(n^2)$.

   (b) [10 points] What is the time complexity of the recursive code given above?

   $T(n) = T(n/2) + T(n/2) + 1$. Thus $T(n) = \Theta(n)$.

   (c) [10 points] What is the time complexity of a dynamic programming algorithm to compute `george(n)`?

   $G(0), G(1), G(2), \ldots G(n)$ are computed, which takes $\Theta(n)$ time.

(d) [10 points] What is the space complexity of a computation of `george(n)` using memoization?

$\Theta(\log n)$, since only values of george$(n/2^t)$ for $t \leq \log_2 n$ are needed.

11. [20 points] Walk through mergesort with the array given below.

```
          VJATNLDQMEFSPWGL
     VJATNLDQ          MEFSPWGL
  VJAT     NLDQ     MEFS     PWGL
VJ  AT   NL  DQ   ME  FS   PW  GL
V J A T N L D Q M E F S P W G L
JV  AT   LN  DQ   EM  FS   PW  GL
 AJTV     DLNQ     EFMS     GLPW
    ADJLNQTV          EFGLMPSW
         ADEFGJLLMNPQSTVW
```

12. [20 points] Write pseudcode for the simple coin-row problem we discussed in class. Your are given a row of n coins of various values. The problem is to select a set of coins of maximum total value, subject to the condition that no two adjacent coins are selected. Your code should identify the coins which are selected.

    We will number the coins from 1 to n. Assume each coin has a positive value. Let $X(i)$ be the value of the i^{th} coin. We call a subset of the coins *legal* if it contains no two adjacent coins. That is, it cannot contain both the i^{th} coin and the $(i+1)^{\text{st}}$ coin for any i. The goals is to find the legal subset of maximum total value.

    We can assume that any two consecutive coins of a legal subset are either 2 or 3 apart, since, if coins i and $i+d$ are in the subset for $d \geq 4$, we can insert the coin $i+2$ and increase the value of the subset.

    The problem reduces to the problem of finding the maximum weight path in a weighted directed acyclic graph (dag, for short) G, where the vertices are weighted (instead of the arcs). The set of vertices is $V = \{1, 2, \ldots n\}$ where i has weight $X(i)$, and the set of arcs is the set of all (i, j) where either $j = i+2$ or $j = i+3$.

    Let $W(i)$ be the weight of the maximum weight path in G which ends at i, for $i \geq 3$, let $B(i)$ be the backpointer, the second to the last coin in that path. Our dynamic program computes both W and B.

    W(1) = X(1)
    W(2) = X(2)
    W(3) = X(1)+X(3)
    B(3) = 1
    for $i = 4$ to n

{
if($W(i-2) > W(i-3)$)
$B(i) = i - 2$
else
$B(i) = i - 3$
$W(i) = X(i) + W(B(i))$
}

The maximum value of any legal subset is $\max(W(n), W(n-1))$, and that subset can be recovered using the backpointers.

13. [20 points] Write pseudocode for a function `float power(float x, int n)` that returns x^n. You may assume that $x \neq 0$, but your code must work for any integer n. It is not necessary to use the algorithm given in class; use any algorithm that works.

We model our code on the code for multipication from Problem 8. We simply change addition to multipication, and multiplication to exponentiation. The additive identity 0 is replaced by the multiplicative identity 1. The loop invariant of the while loop is $x^n = y^m * z$

```
float power(float x, int n)
 {
  if(n == 0) return 1;
  else if(n < 0) return power(1/x,-n);
  else
   {
    float z = 1.0;
    float y = x;
    int m = n;
    while(m > 0)
     {
      if(m%2) z = z*y;
      m = m/2;
      y = y*y;
     }
    return z;
   }
 }
```

14. [20 points] The following code is used as a subroutine for both quicksort and select. Assume A[n] is an array of integers. For simplicity, we assume that no two entries of A are equal. Write a loop invariant for the while loop.

```
int pivot = A[0];
int lo = 0;
int hi = n-1;
while(lo < hi)
 {
  if(A[lo+1] < pivot) lo++;
  else if(A[hi] > pivot) hi--;
  else swap(A[lo+1],A[hi]);
 }
```

The loop invariant is the conjuction of three statements:

lo $\leq$ hi and $A[i] \leq$ pivot for all 0 $\leq$ i $\leq$ lo and $A[i] >$ pivot for all hi $< i \leq n$.