1. True or False. [5 points each]
   (a) T A binary search tree is a search structure.
   (b) T A minheap is a priority queue.
   (c) F A good programmer would never store data in an unordered list.

2. Fill in the blanks.
   (a) [10 points] Θ(n) What is the asymptotic complexity of merging two sorted lists, each of length $n$? Use $\Theta$ notation.
   (b) [10 points] A stack is a priority queue in which the most recently inserted item has priority.
   (c) [10 points] Dijkstra’s algorithm does not allow the weight of any arc to be negative.
   (d) [10 points] binary search is a divide and conquer algorithm which implements the operator find for an ordered list.
   (e) [10 points] fetch and store are operators of the ADT array.
   (f) [10 points] The items in a priority queue represent unfulfilled obligations
   (g) [10 points] The worst case number of comparisons of any comparison/exchange sorting algorithm is $\Omega(n \log n)$.
   (h) [10 points] radix sort is a sorting algorithm which does not use the comparison/exchange model of computation.
   (i) [20 points] quicksort and mergesort are divide-and-conquer sorting algorithms.
   (j) [20 points] What is the asymptotic time complexity for the Bellman-Ford algorithm on a weighted directed graph with $n$ vertices and $m$ edges, where, for some number $p$ and for every vertex $x$, the least weight path from the source to $x$ has no more than $p$ edges? $O(mp)$
3. [20 points] Walk through Dijkstra’s algorithm for the following weighted directed graph.
4. [20 points] Write pseudocode for the Floyd–Warshall algorithm. We assume that \( W[i,j] \) is the weight of the edge from \( i \) to \( k \), if there is one. if there isn’t one, we assume \( W[i,j] = \infty \).

\[
\text{for all } i \text{ and all } j \\
\{ \\
V[i,k] = W[i,j]; \\
\text{back}[i,j] = i; \\
\}
\]

\[
\text{for all } i \ V[i,i] = 0; \\
\text{for all } j \\
\text{for all } i \text{ and all } k \text{ in either order} \\
\{ \\
\text{temp} = V[i,j] + V[j,k]; \\
\text{if} (\text{temp} < V[i,k]) \\
\{ \\
V[i,k] = \text{temp}; \\
\text{back}[i,k] = \text{back}[j,k]; \\
\}
\}
\]

5. [20 points] What is the purpose of the function \texttt{george} below? Multiplication

Give a loop invariant for the main loop.

\[
rslt + ym = xn
\]

\[
\text{int george(int x, int n)} \\
\{ \\
\text{// input condition: } n \geq 0 \\
\text{int } y = x; \\
\text{int } m = n; \\
\text{int } rslt = 0; \\
\text{while}(m > 0) \\
\{ \\
\text{if}(m \% 2) \ // \text{ m is odd} \\
\{ \\
\text{m} = m-1; \\
rslt = rslt + y; \\
\}
\text{else} \\
\{ \\
\text{m} = m/2; \\
y = y+y; \\
\}
\text{cout} \ll rslt; \\
\}
\}
\]
6. Name each of these algorithms. 10 points each.

(a) **quicksort** Pick an element \( P \) from a set \( S \), then partition \( S \) into two parts: those items which are less than \( P \) and those greater than \( P \). Recursively sort each part, and combine them to form a sorted list.

(b) **mergesort** Divide a set \( S \) arbitrarily into two equal parts. Recursively sort each part, then combine the two sorted parts to obtain a sorting of \( S \).

(c) **binary search** Given a sorted set \( S \) and an item \( x \), you need to determine whether \( x \in S \). Pick one element, say \( m \), out of \( S \). If \( m = x \), you are done. If \( m < x \), discard \( m \) and all items of \( S \) which are greater than \( m \), while if \( x > m \), discard \( m \) all items which are all items of \( S \) which are less then \( m \). Keep doing this until you either find \( x \) or you have discarded all items of \( S \).

(d) **treesort** Given a set \( S \), create an empty binary search tree \( T \). Insert the items of \( S \) into \( T \) one at a time. Finally, visit and print the items of \( T \) in left-to-right order, also called inorder.

(e) **selection sort** Given a set \( S \), delete the least element of \( S \) and print it. Then delete the least remaining element of \( S \) and print it. Keep going until you have deleted and printed all elements of \( S \).

(f) **linear search** Look at each item in a list, starting at the head. If one of the items is equal to \( X \), then stop and report that you have found \( X \). If you reach the end of the list without finding \( X \), report that \( X \) is not in the list.

7. [20 points] Execute heapsort with input file ASQWFGKZ. Use the array below. Add additional rows if needed.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S</td>
<td>Q</td>
<td>W</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Z</td>
</tr>
<tr>
<td>A</td>
<td>S</td>
<td>Q</td>
<td>Z</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>W</td>
</tr>
<tr>
<td>A</td>
<td>Z</td>
<td>Q</td>
<td>S</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>W</td>
</tr>
<tr>
<td>A</td>
<td>Z</td>
<td>Q</td>
<td>W</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>S</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>Q</td>
<td>W</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>S</td>
</tr>
<tr>
<td>Z</td>
<td>W</td>
<td>Q</td>
<td>A</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>S</td>
</tr>
<tr>
<td>Z</td>
<td>W</td>
<td>Q</td>
<td>S</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>W</td>
<td>Q</td>
<td>S</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Z</td>
</tr>
<tr>
<td>W</td>
<td>A</td>
<td>Q</td>
<td>S</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Z</td>
</tr>
<tr>
<td>W</td>
<td>S</td>
<td>Q</td>
<td>A</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Z</td>
</tr>
<tr>
<td>K</td>
<td>S</td>
<td>Q</td>
<td>A</td>
<td>F</td>
<td>G</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>S</td>
<td>K</td>
<td>Q</td>
<td>A</td>
<td>F</td>
<td>G</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>G</td>
<td>K</td>
<td>G</td>
<td>A</td>
<td>F</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>Q</td>
<td>K</td>
<td>G</td>
<td>H</td>
<td>F</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>F</td>
<td>K</td>
<td>G</td>
<td>A</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>K</td>
<td>F</td>
<td>G</td>
<td>A</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
<td>A</td>
<td>K</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>G</td>
<td>K</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>G</td>
<td>K</td>
<td>Q</td>
<td>S</td>
<td>W</td>
<td>Z</td>
</tr>
</tbody>
</table>

4
8. [20 points] The following code correctly computes the $n^{th}$ Fibonacci number. However, it is not a good idea to use this code. Why not? How would you solve the same problem differently?

```c
int fibonacci(int n)
{
    assert(n > 0)
    if(n <= 2) return 1;
    else return fibonacci(n-2) + fibonacci(n-1);
}
```

It will take exponential time. Use dynamic programming or memoization instead.

9. [20 points] Find the strong components of the directed graph shown below, using the DFS method in our textbook.