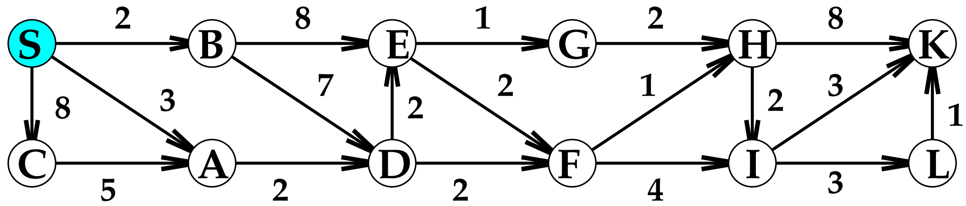
University of Nevada, Las Vegas Computer Science 477/677 Spring 2022

Answers for Examination March 23, 2022

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] $\Theta(n)$ What is the asymptotic complexity of merging two sorted lists, each of length n? Use Θ 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.



	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7	8	8	~~11~~ 10	~~16~~ 13	13
back		S	S	S	~~B~~ A	~~B~~ D	D	E	F	~~F~~ H	~~H~~ I	I

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 assum W[i,j] $= \infty$.

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

5. [20 points] What is the purpose of the function `george` below? Multiplication

Give a loop invariant for the main loop.

rslt $+ym = xn$

```
int george(int x, int n)
 {
  // input condition: n >= 0
  int y = x;
  int m = n;
  int rslt = 0;
  while(m > 0)
   {
    if(m%2) // m is odd
     {
      m = m-1;
      rslt = rslt + y;
     }
    else
     {
      m = m/2;
      y = y+y;
     }
    cout << 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. Recorsively sort each part, and combine them to form a sorted list.

   (b) **mergesort** Divide a set S abitrarily 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 then 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.

1	2	3	4	5	6	7	8
A	S	Q	W	F	G	K	Z
A	S	Q	Z	F	G	K	W
A	Z	Q	S	F	G	K	W
A	Z	Q	W	F	G	K	S
Z	A	Q	W	F	G	K	S
Z	W	Q	A	F	G	K	S
Z	W	Q	S	F	G	K	A
A	W	Q	S	F	G	K	**Z**
W	A	Q	S	F	G	K	**Z**
W	S	Q	A	F	G	K	**Z**
K	S	Q	A	F	G	**W**	**Z**
S	K	Q	A	F	G	**W**	**Z**
G	K	G	A	F	**S**	**W**	**Z**
Q	K	G	H	F	**S**	**W**	**Z**
F	K	G	A	**Q**	**S**	**W**	**Z**
K	F	G	A	**Q**	**S**	**W**	**Z**
A	F	G	**K**	**Q**	**S**	**W**	**Z**
G	F	A	**K**	**Q**	**S**	**W**	**Z**
A	F	**G**	**K**	**Q**	**S**	**W**	**Z**
F	A	**G**	**K**	**Q**	**S**	**W**	**Z**
A	**F**	**G**	**K**	**Q**	**S**	**W**	**Z**
A	**F**	**G**	**K**	**Q**	**S**	**W**	**Z**

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?

```
\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.