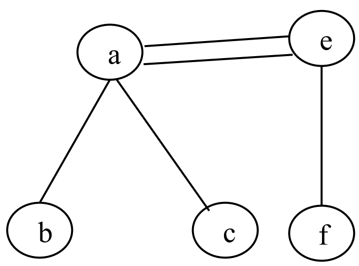
CST 370 – Spring A 2020
Homework 2

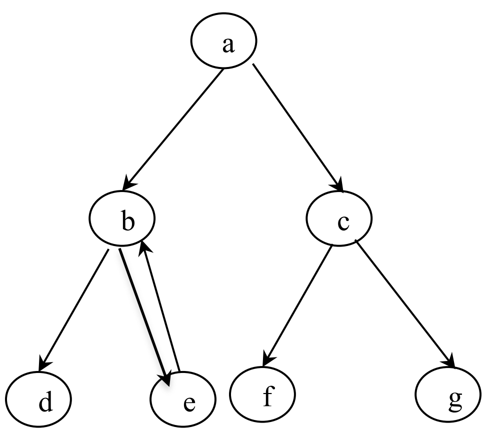
Name: Sara Kazemi

Class ID: 4444

1. (5 points) (a) Based on our textbook's definition, is this a graph? (True/False) Explain to support your answer. **False. Our textbook's definition of an undirected graph disallows multiple edges between the same vertices. This is violated below as there are two edges between vertex a and vertex e.**



(b) Based on our textbook's definition, is this a graph? (True/False) Explain to support your answer. **True. According to our textbook's definition, the following is a directional graph (digraph) because the edges (b, e) and (e, b) are not the same. The edge represented by (b, e) leaves b and enters e, while the edge represented by (e, b) leaves e and enters b. This relationship is indicated by directional arrows, which is what the above example lacked.**

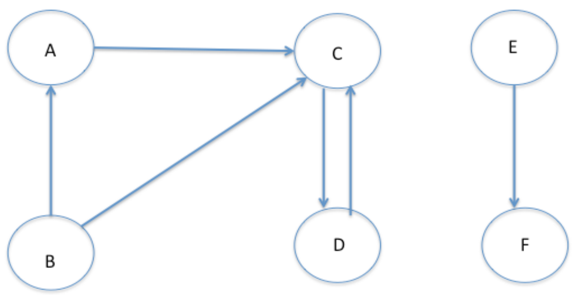


2. (10 points) Assume that you should search a number in a list of *n* numbers. How can you take advantage of the fact that **the list is known to be sorted**? Give separate answers for the following two cases.

(a) A list represented in an array. **Because the array is sorted, we can utilize a binary search.**

(b) A list represented in a linked list. **If the linked list is sorted, we should stop searching once we reach an element that is either greater than or equal to the search key.**

3. (5 points) Represent the following graph in the adjacency list as you learned in the class. Note that there are **six vertices** (= A, B, C, D, E, and F) in the graph.



A	→ **C** →
B	→ **A** → **C**
C	→ **D**
D	→ **C**
E	→ **F**
F	

4. (5 points) Assume a binary tree with six vertices such as v1, v2, v3, v4, v5, and v6. Determine the maximum number of edges possible in the tree.
The maximum possible edges in a binary tree with 6 vertices can be determined by:

$$|E| = |V| - 1$$

The maximum number of edges in this tree is 5.

5. (5 points) (a) If your program takes *n*log n* time and your classmate's program takes n^2 time, whose program is faster? Pick one between "You" and "Your Classmate". **My program is faster.**

(b) If your program takes *log n* time and your classmate's program takes *constant* time, whose program is faster? Pick one between "You" and "Your Classmate". **My classmate's program is faster.**

6. (10 points) Consider the following algorithm.

```
Algorithm Compute (A[0.. n – 1])
1.  num1 ← A[0];
2.  num2 ← A[0]
3.  i ← 1
4.  while i < n do
5.      if A[i] < num1
6.          num1 ← A[i];
7.      if A[i] > num2
8.          num2 ← A[i];
9.      i ← i + 1
10. return (num2 – num1);
```

(a) Present the basic operation of the algorithm. When you present the basic operation, you should indicate the line number of the basic operation clearly. **The basic operation < is present on Line 4 as this conditional determines how many times we compare an element in the array A to the values num1 and num2.**

(b) Present the time complexity category of the algorithm among the eight most popular time complexity categories we covered in the lecture. ***O(n) since it's dependent on the length n of the array.***

7. (10 points) Consider the following algorithm

```
1. Algorithm Mystery(n)
2. // Input: A nonnegative integer n
3. S ← 0
4. for i ← 1 to n do
5.     k ← i * i
6.     S ← S + k
7. return S
```

(a) What does this algorithm compute? **Sums together the squares from 1 to n.**
(b) What is its basic operation? **The basic operation is present on Line 4. We can infer is the basic operation is < since this represents a loop that will execute for all values from 1 to n.**
(c) Present the time complexity category of the algorithm among the eight most popular time complexity categories we covered in the lecture. ***O(n)***

8. (25 points) Write a C++ program called **sieve.cpp** that implements the **sieve of Eratosthenes** algorithm in our textbook (page 6 ~ 7). **See attached file sieve.cpp**

Grading guide and test cases for the sieve.cpp:

- Missing head comment elements (= Title, Abstract, ID, Name, and Date).
- Compilation failed → It will be very serious.
- Your program should use the dynamic memory for array(s).
- Test case 1: Input number 2 → 2
- Test case 2: Input number 10 → 2, 3, 5, 7
- Test case 3: Input number 25 → 2, 3, 5, 7, 11, 13, 17, 19, 23
- Test case 4: Input number 101 → 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101

9. (25 points) Write a C++ program "palindrome.cpp" that reads a string of characters from user and determines if the string is a palindrome or not. For the program, you should store each character of the string in an array (use **dynamic memory** to create array(s)) and follow the algorithm described in the lecture. **See attached file palindrome.cpp**

Palindrome is a string that reads the same from both the beginning and the end. Here are sample test cases:

(1) Input string 1: racecar
Output: Yes, it's a palindrome
(2) Input string 2: abcdefghijihgfedcba
Output: Yes, it's a palindrome
(3) Input string 3: CSUMB
Output: No, it's not a palindrome