CST 370 – Spring A 2020
Homework 4

Name:______________________

Class ID: _________________

<u>How to turn in?</u>

- Write your answer to the questions 1 to 5, and submit it on the iLearn. You can submit the file in PDF format. Don't forget to write your name and class ID number at the beginning of the file.
- For Questions 6, 7, and 8, you should submit your C++ source files on the iLearn.
- Thus, you have to submit four files (one PDF file and three C++ source file) on the iLearn.
- Note that the due date is 11:55(PM). This is the iLearn's timestamp, not your submission time. Since there could be a long delay between your computer and iLearn, you should **submit early**.

1. (10 points) Consider the Master Theorem as we covered in the lecture.

$$T(n) = aT(n/b) + f(n) \quad \text{where } f(n) \in \Theta(n^d), \quad d \geq 0$$

<u>Master Theorem</u>: If $a < b^d$, $\quad T(n) \in \Theta(n^d)$
If $a = b^d$, $\quad T(n) \in \Theta(n^d \log n)$
If $a > b^d$, $\quad T(n) \in \Theta(n^{\log_b a})$

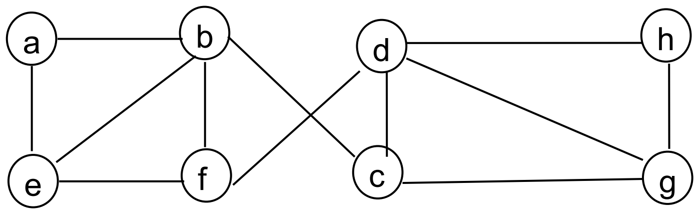
Based on the theorem, determine the time efficiency of the following formula T(n).

(a) $T(n) = 2 * T(n/2) + n^4$

(b) $T(n) = 16 * T(n/4) + n^2$

You need to indicate the values for a, b, and d. Also, indicate which case of the Master Theorem applies.

2. (10 points) Consider the following graph.



Starting at vertex ***a***, traverse the graph using the **depth-first search** (DFS) algorithm. After that, you should present a **mark array** and a **depth-first search tree** with only tree edges. For the algorithm, you have to use **our convention** of the class (= **ascending order for the alphabetical characters**).

[**Note**: If it is difficult to draw the results on the word file, draw them on paper by hand. Then take a picture of the paper and attach the picture here.]

3. (20 points) (a) Assume that you are going to develop an algorithm to find the position of the largest number in an array with *n* integer numbers using **a divide-and-conquer technique**. For example, if your algorithm has an input array such as **1, 3, 11, 7, 5, 6, 4, 9,** your algorithm should return 2 because the index 2 has the largest number (= 11) in the array. Describe the basic idea of your divide-and-conquer algorithm **in English**.
[**Hint**: Read the Google document at https://goo.gl/ySZavW and pay attention to the sample function "sum_div_N_conq()" in the document. You can get the basic idea of the problem from the function.]

(b) Based on the basic idea of (a), **write a pseudocode** of your divide-and-conquer algorithm. Note that the array index of your algorithm should start from zero.

(c) Determine the time complexity of your algorithm. Set up a recurrence relation and apply the Master Theorem to calculate the time complexity. So, you should provide the values of symbols "a", "b", and "d". And then provide the time efficiency of your algorithm

4. (5 points) Using the mergesort pseudocode in our textbook, sort the numbers *13, 22, 57, 99, 39, 64, 57, 48, 70* in the ascending order. You should describe your answer as Figure 5.2 of our textbook. Note that our textbook's example has 8 numbers. However, **this problem has 9 numbers**. Thus, you should read the pseudocode carefully and identify how it sorts the 9 numbers. *Note that the second "copy" statement of the Mergesort algorithm in the textbooks has a typo: for odd number of elements, the second copy should have one more element than the first copy.*

[**Note**: If it is difficult to draw the operation on the word file, draw it on paper by hand. Then take a picture of the paper and attach the picture here.]

5. (5 points) Consider the quicksort algorithm covered in the class. Present the **first partitioning operation** for the list "**6 10 13 5 8 3 2 11** ", You should use the first number, 6, as a pivot for the partitioning. In your answer, you have to present the indexes i and j clearly.

[**Note**: If it is difficult to draw the operation on the word file, draw it on paper by hand. Then take a picture of the paper and attach the picture here.]

6. (15 points) Write a program called ***select-sort-k.cpp*** that modifies the selection sort algorithm to sort the first *k* smallest elements of the array with *n* elements (k <= n, the value of k will be entered by the user). Your program should read the integer numbers from an input file, store them in an array (dynamic), and display the k smallest elements in sorted order. Your algorithm must run in O(n*k) time. When you

write the program, don't forget to include "Title", "Abstract", "ID (A four-digit number)", "Name", and "Date".

Sample input file:

This is the content of **t1.txt**. Note that the **first line** (10) indicates the number of integers in the file.

```
10
4
6
8
15
20
22
10
3
9
2
```

Sample Run 1:

```
Enter a file name: ./t1.txt

Enter the value k: 4

Output: 2, 3, 4, 6
```

Sample Run 2:

```
Enter a file name: ./t1.txt

Enter the value k: 6

Output: 2, 3, 4, 6, 8, 9
```

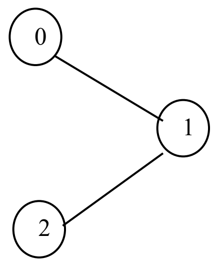
7. (20 points) Write a C++ program called **BFS.cpp** that implements the ***Breadth-First Search (BFS) algorithm***. Your program should read an input file name and a starting node from a user. After that, your program should display the list of vertices visited. In the problem, you can assume that the number of vertices in the input file is less than or equal to 25. When you write the program, don't forget to include "Title", "Abstract", "ID (A four-digit number)", "Name", and "Date".

[**Hint**: Since the maximum number of vertices is less than or equal to 25, you can use a simple array-based queue for the BFS algorithm. Or you can use a STL queue library. For more info, refer to http://www.cplusplus.com/reference/queue/queue/pop/]

Input file format: This is a sample input file called `t1.txt`.

```
3
0 1 0
1 0 1
0 1 0
```

The first line (= 3 in the example) indicates that there are three vertices in the graph. For the homework, we can assume that the first vertex starts from the number 0. Thus, **t1.txt** describes a graph like below:



One blank space is used to delimiter the data. Note that there's no blank space at the end of each line. **If your program does not read the file properly, your program will get no credit.**

This is a sample run:

```
Enter filename: ./t1.txt
Enter a start vertex: 0
BFS order: 0 -> 1 -> 2
```

In the program, your program has to **follow our convention (= ascending order)**.

This is another sample input file called **`t2.txt`**.

```
5
0 0 1 0 0
0 0 1 0 0
1 1 0 1 1
0 0 1 0 0
0 0 1 0 0
```

This is a sample run:

```
Enter filename: ./t2.txt
Enter a start vertex: 0
BFS order: 0 -> 2 -> 1 -> 3 -> 4
```

8. (15 points) Write a C++ program called **assignment.cpp** that solves the *Assignment Problem* in Chap 3.4. Your program should read the assignment costs of each person from a user and determine the best assignment. In the program, you can assume that the number of all jobs is less than 15. To get the full credits, your program should display all combinations of assignments. However, the sequence of combinations to be displayed on the screen is not important.

This is a sample run:

```
Enter number of jobs: 2
Enter assignment costs of 2 persons:
Person 1: 2 5
Person 2: 4 6
Output:
Permutation 1: <1, 2> => total cost: 8
Permutation 2: <2, 1> => total cost: 9

Solution: <1, 2> => total cost: 8
```

This is another sample run

```
Enter number of jobs: 3
Enter assignment costs of 3 persons:
Person 1: 2 5 3
Person 2: 4 6 9
Person 3: 7 2 4

Permutation 1: <1, 2, 3> => total cost: 12
Permutation 2: <1, 3, 2> => total cost: 13
Permutation 3: <2, 1, 3> => total cost: 13
Permutation 4: <2, 3, 1> => total cost: 21
Permutation 5: <3, 1, 2> => total cost: 9
Permutation 6: <3, 2, 1> => total cost: 16

Solution: <3, 1, 2> => total cost: 9
```