COMP 3270 Assignment 4 5 problems 50 points 10% Credit

Due before 11:59 PM Thursday November 4

Instructions:

1. This is an individual assignment. You should do your own work. Any evidence of copying will result in a zero grade and additional penalties/actions.
2. Enter your answers in this Word file. Submissions must be uploaded **as a single file** (Word or PDF preferred, but other formats acceptable as long as your work is LEGIBLE) to Canvas before the due date and time. Don't turn in photos of illegible sheets. **If an answer is unreadable, it will earn zero points.** Cleanly handwritten submissions (print out this assignment and write answers in the space provided, with additional sheets used if needed) scanned in as PDF and uploaded to Canvas are acceptable.
3. **Submissions by email or late submissions (even by minutes) will receive a zero grade.** No makeup will be offered unless prior permission to skip the assignment has been granted, or there is a valid and verifiable excuse.
4. Think carefully; formulate your answers, and then write them out concisely using English, logic, mathematics and pseudocode (no programming language syntax).

1. (15 points) **Binary Heap**

Max-Heap-Increase-Key(A[1...n]: array of number, i: int 1≤i≤n, key)

```
1  if key < A[i]
2    then print "new key is smaller than current key"
3  A[i] = key
4  parent = floor(i/2)
5  while i > 1 and A[parent] < A[i]
6          temp = A[i]
7          A[i]= A[parent]
8          A[parent] = temp
9          i = parent
10         parent = floor(i/2)
```

Show that the complexity of this algorithm is $O(\log_2 n)=O(\lg n)$ by developing and stating its T(n) in which the largest n-term is a lgn term. Do this by filling in the table and blanks below. Some entries are pre-filled. Cost of the floor operation = 1

Step#	Cost of single execution	Exact # of times executed	Total cost of this step = column 1 * column 2
1	**5**	1	**5**
2	1	1	1
3	**4**	**1**	**4**
4	**4**	**1**	**4**
5	**10**	At most **lgn + 1** times	**10(lgn + 1)**
6	**4**	At most **lgn** times	**4lgn**
7	**6**	At most **lgn** times	**6lgn**
8	**4**	At most **lgn** times	**4lgn**

9	**2**	At most **lgn** times	**2lgn**
10	**4**	At most **lgn** times	**4lgn**

Sum of last column: 5 + 1 + 4 + 4 + 10(lgn + 1) + 4lgn + 6lgn + 4lgn + 2lgn + 4lgn = 24 + 30lgn
Sum the last column and simplify to obtain T(n) < **24 + 30lgn**

2. (14 points) **Quick Sort**

Come up with an input of size 7 that will:

(a) produce the best case partitions in every recursive call of Quick Sort based on the Quick Sort and Partition algorithms that are given in the lecture slides.

A=

10	**30**	**20**	**40**	**50**	**60**	**35**

(b) produce the worst case partitions in every recursive call of Quick Sort based on the Quick Sort and Partition algorithms that are given in the lecture slides.

A=

70	**60**	**50**	**40**	**30**	**20**	**10**

3. (5 points) **Counting Sort**

The Counting Sort algorithm can be used to sort integers in the range i-j, i<j and i>0 by pre-processing the input array A so that the algorithm can be applied to it as is with no modifications and then post-process the output array B to recover the original input in the sorted order. Explain in English what this will entail:

(a) What is the pre-processing on A that can be done so that the algorithm can work with no modifications?

Since the algorithm works for an array with input elements in the range from 0 to k for some integer k, the pre-processing on A would have to modify the elements in A to fit such a range. If array A originally has the range i – j , i < j and i > 0 the pre-processing would include performing some operation on each element such that the new range was 0 – k. One way to do this would be to subtract the smallest value in the array from every element. So if the array was [2, 5, 3, 4] (i = 2, j = 5), then the pre-processing would subtract 2 (the smallest element) from all elements in the array. The result would be [0, 3, 1, 2]. This array's range is now 0 – 3 and therefore fits the range specifications (0 – k) for the algorithm.

(b) What is the value of k in this case (the algorithm requires prior knowledge of the input range 0-k)?

The value of k would be the largest element in the array. After the pre-processing this value will be j – i (the largest value from the initial range minus the smallest value from the initial range).

(c) What is the post-processing on B that can be done so that the algorithm can work with no modifications?

Since pre-processing only involved subtracting the smallest element from the original array, all that must be done to get back the original array in sorted order is to add the value of i (the smallest element) back to the array.

4. (7 points) **Radix Sort**

If Radix Sort is used to sort an array of words alphabetically, and the input array is A=

CATS	BATS	BITS	PINE	DIG< >	BORE	DIM< >

show the array after each pass of the outer loop of the algorithm completes. < > is a single blank character that is used to pad words with less than 4 characters and it appears before the letter A in alphabetic ordering.

A after the first execution of the loop=

DIG<>	**DIM<>**	**PINE**	**BORE**	**CATS**	**BATS**	**BITS**

A after the second execution of the loop=

DIG<>	**DIM<>**	**PINE**	**BORE**	**CATS**	**BATS**	**BITS**

A after the third execution of the loop=

CATS	**BATS**	**DIG<>**	**DIM<>**	**PINE**	**BITS**	**BORE**

A after the fourth execution of the loop=

BATS	**BITS**	**BORE**	**CATS**	**DIG<>**	**DIM<>**	**PINE**

5. (9 points) **Bucket Sort**

If length(A)=10 then numbers in the input array in the range [0,0.1) will all go to bucket 0, numbers in the input array in the range [0.1,0.2) will all go to bucket 1, numbers in the input array in the range [0.2,0.3) will all go to bucket 2, numbers in the input array in the range [0.3,0.4) will all go to bucket 3, numbers in the input array in the range [0.4,0.5) will all go to bucket 4, numbers in the input array in the range [0.5,0.6) will all go to bucket 5, numbers in the input array in the range [0.6,0.7) will all go to bucket 6, numbers in the input array in the range [0.7,0.8) will all go to bucket 7, numbers in the input array in the range [0.8,0.9) will all go to bucket 8, and numbers in the input array in the range [0.9,1.0) will all go to bucket 9. If length(A)=9 then list the range of input numbers that will go to buckets 0…8. State your answers with two decimal digit precision.

Numbers in the input array in the range [**0.00, 0.11**) will all go to bucket 0
Numbers in the input array in the range [**0.11, 0.22**) will all go to bucket 1
Numbers in the input array in the range [**0.22, 0.33**) will all go to bucket 2
Numbers in the input array in the range [**0.33, 0.44**) will all go to bucket 3
Numbers in the input array in the range [**0.44, 0.56**) will all go to bucket 4 ***without rounding: [0.44, 0.55)**
Numbers in the input array in the range [**0.56, 0.67**) will all go to bucket 5 ***without rounding: [0.55, 0.66)**
Numbers in the input array in the range [**0.67, 0.78**) will all go to bucket 6 ***without rounding: [0.66, 0.77)**
Numbers in the input array in the range [**0.78, 0.89**) will all go to bucket 7 ***without rounding: [0.77, 0.88)**
Numbers in the input array in the range [**0.89, 1.00**) will all go to bucket 8 ***without rounding: [0.88, 1.00)**