Name: Brian Sherif Nazmi

AaDS HW6

Problem 1:

a) Input sequence: <9, 1, 6, 7, 6, 2, 1>

sort using counting sort

Input array:

9	1	6	7	6	2	1

Counter array:

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Create a counter array size of 10 as the largest input value is 9. set them all to zero.

Now create a iterator, which passes over the input array and with every value it sees it adds a 1 to the values position in the array.

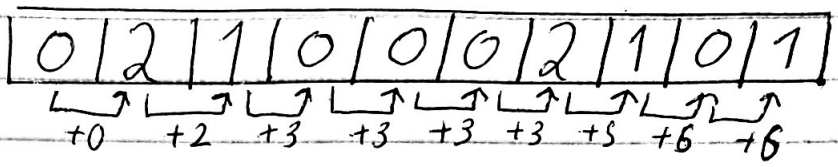
0	1	2	3	4	5	6	7	8	9
0	2	1	0	0	0	2	1	0	1

There is a 2 here as we have 2 '1' in our input array

because we don't have an 8 in our input

We must now modify our counter array by taking each element and adding it to the one on its right hand side

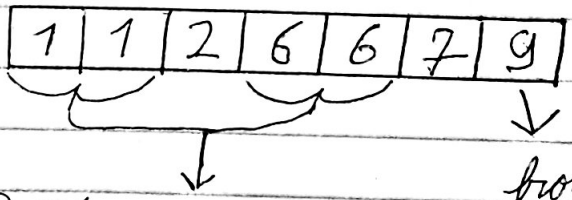
counter array:



Modified counter array:

0	1	2	3	4	5	6	7	8	9
0	2	3	3	3	3	5	6	6	7

Now create a new array with length of input array. Then go thru input array taking each value in input array as the location in the modified counters array. Then take the value in that location of the counters array as the location of the checked value in the input array in the newly created array. Then subtract 1 from counter



Both are in correct spot due to the subtraction we do at the end

from counter is in position 7

Sorted.

Problem 1

b) input sequence $\langle 0,9; 0,1; 0,6; 0,7; 0,6; 0,2; 0,1 \rangle$

input array

0,9	0,1	0,6	0,7	0,6	0,2	0,1

we have 7 elements in input array.
Now create a 2d array of which column size is 7 and row size expands with inputs.

0: 0,1 ; 0,1
1: 0,2
2
3
4: 0,6 ; 0,7 ; 0,6
5
6: 0,9
7

Now multiply each input by 7 and round it to the closest possible int:

$0,9 \cdot 7 = 6,3 \approx 6$
$0,1 \cdot 7 = 0,7 \approx 0$
$0,6 \cdot 7 = 4,2 \approx 4$
$0,7 \cdot 7 = 4,9 \approx 4$
$0,6 \cdot 7 = 4,2 \approx 4$
$0,2 \cdot 7 = 1,2 \approx 1$
$0,1 \cdot 7 = 0,7 \approx 0$

I then use this rounding as their column address.

After sorting them into Buckets, implement insertion sort which arranges elements in the Buckets.

After that create new array with length of input and iterate there every column inserting them into array

0,1	0,1	0,2	0,6	0,6	0,7	0,9

Sorted

Problem 1

c) Given an input of length n; The worst case scenario would be if Bucket sort decided to put them all in 1 Bucket (Assuming because all entries are very close together that when rounded give the same integer.

Over that if they are placed in the Bucket in reversed order. This will also be the worst case for insertion sort. (which is implemented to sort the buckets. This gives a worst case scenario of insertion sort $O(n^2)$ to Bucket sort.

worst case $O(n^2)$

because it will have to iterate thru array

$O(2n) + O(n^2) = O(n^2)$

---

g) (inspired by Insertion sort)

```
Struct Point A[]                    // Struct point containing coordinates
~~Int B[]~~ Int n = length.(A);
Int B[n];
for(int i=0; i<n; i++){
B[i] = sqrt((A.x)^2+(A.y)^2)        // B[i] = Distance
for(int K=n; K>1 && B[K]<B[K-1]; K--]
    swap B[K, K-1]
    swap A[K, K-1]
} now A & B are sorted
end
```

Problem 1

c)

```
int arr [1...n] ;          // Assume that
                              this is input
K = length(arr);
int counter [K] = {0}      // create counter inspired
                              by count sort
for(int i=0; i<n; i++){
    counter[arr[i]] = + 1 ; // each time element
}                             of arr is iterated add 1
                              to address in counter
for(int j=1; j<K; j++){
    counter(j) = counter[j] + counter[j+1] ;
}                          // add previous element
                           to
```

Problem 2

b) Since code not working I found a version online that states

if every element in the array of size n (n being number of inputs. we need roughly $\log_{10} n$ digits in the decimal system.

Each call to counting sort takes time $O(n+10) = O(n)$

because each digit has 10 K possibilities

total time = $O(n \log_{10} n) = O(n \log n)$

Radix's space complexity is bound by the sort it uses to sort each radix in best case (counting sort)

this means it will use counting sort space complexity $O(n+K)$

c) I believe the answer lies in using Radix sort. With count sort implemented in it. Divided onto 3 Buckets and run count sort on each.