Homework 8 by Timofei Podlorytov

8.1

(a-b) are in the cpp file

c) here the idea is to modify the count algorithm and stop at the place where we count prefix sums. Then we subtract the prefix sums of the range and get the answer. Also, I added some if clauses in case of the a and b falling out of range.

```
CountSort(A,n,k,a,b):
        for i := 1 to k do
                C [ i ] := 0
        for j := 1 to n do
                C [ A [ j ]] := C [ A [ j ]] + 1
        for i := 2 to k do
                C [ i ] := C [ i ] + C [ i – 1]
        if b>k: //these are the safety measures
                b=k;
        if a<=0:
                return C[b];
        return C[b]-C[a-1]//we use the prefix sums
```

d)is implemented in the corresponding file

e)

worst case for bucket sort is when the condition is not satisfied and all the values are in the same bucket resulting in no benefit in terms of time. As a result the worst case time complexity would be the worst case time complexity for insertion sort which is quadratic. Thus our worst case is a decreasing sequence where all the values are close to each other and end up in the same bucket.

$O(n^2)$

8.2

a)The implementation is in the cpp file.

b)

we call the sort function twice and end the call when we go through all positions, approximately we split it in half, but all in all the sum is n

1 n =n

2 n1+n2=n

…

pos-1 n1+n2+n3.. n_k=n

pos n1+n2+n3.. n_{2k}=n

the sum is n*pos→ the time complexity is $O(n*pos)$, where the pos stands for the number of positions/bits we need to look through