Lab 4

1) $\boxed{\frac{1}{p}} \quad P(2) = \frac{1}{10} = \frac{1}{\frac{2}{10}} = 10$

$\frac{k}{p} = 30$.

problem 2:

is the a comparison based algorithm which when run on array containing ~~for~~ 4 elements requires 4 comparisons.

Answer: the possible arrangment of n elements is $n!$, as we have seen from the decision tree for merge sort on 4 elements, the possible ordering ~~for~~ of 4 elements is $4! \rightarrow 24$

$\Rightarrow$ it might be possible to find the leaf node with 4 comparisons that it is not always the case, the total comparison that we require is the depth of longest leaf.

so 4 comparison is not enough but it can be done definitly by 5 comparisons.

problem 3: goofy algorithm

A- it might work if we are really lucky, but there is no garantee that will sort the given input list.

B) the best case if we sort the cards in the first

c) the best Case running time is O(1) which is contant time.

D- the worst case running time is that we keep sorting and sorting and elements are never in a sorted order, this our trial goes as big as ∞ without finding the sorted sequence.

E) it is hard to determine the avege running time for such algorithms but I think the average will still ∞ because there are no garantee when we can find it.

F) no, it is not inversion bound

~~problems~~

problem 4:

A[5, 1, 4, 3, 6, 2, 7, 1, 3]

1) $\frac{3}{4}$ 9 = 6 elements

- the good pivots are 3, 3 and 4.

b) no, they are not in our case, $\frac{n}{2}$ = 4 or 5, but we have only 3 pivots.

problem 5:

Algorithm Sideway (A)

Input: ordered array A from MergeSort Algorithm

output: sideway sorted output.

```
i ← 0
j ← A.length -1
for k ← 0 to A.length -2 do
    if (i != j) then
        newArray[k] ← A[i]
        newArray[k+1] ← A[j]
        i ← i+1
        j ← j-1
        k ← k+2
```

A) The Asymptotic running time would be $O(n \log n)$ (4)
from the merge sort and $O(n)$ additional work

$$O(n \log n + n) = O(n \log n)$$

B) prove that it is impossible to obtain an algorithm to desideway
Sorting of an integer array that's ~~was~~ run faster than
the algorithm in part