Problem 2

b.) The input to the algorithm is an unsorted array of integers $A[a_1 \ldots a_n]$

Pseudo code:

for $i = a_1$ to a_{n-1}

min $= i$

for $j = i + 1$ to n

if $A[j] < A[min]$

min $= j$

Swap $A[i]$ with $A[min]$

Loop invariant:

Before the start of each loop, the following holds

$A[min] \forall A[i \ldots j-1]$ (sorted side of the array

Proof of loop invariant:

Start of program

At the begining of the program, the min is set to the first element in the array, and $j = i+1$ since the algorithm just started $i=[0,1]$ and the min is the first ellment and the only element in the sub array it is hence ~~smaller~~ equal to what is in the sub array $A[min] = A[i \dots j-1]$ and hence loop invariant is true.

Maintenance (Running of code):

Before passing the 2nd for loop; we assume that min = ~~smallest~~ largest element in sub array. largest In the 2nd for loop, there exist 2 possible cases either $A[j] < A[min]$ or does not

case 1: If $A[min] < A[j]$ Then min still represents the ~~smallest~~ largest element in the sorted sub array meaning the loop invariant holds.

case 2: If $A[j] < A[min]$ then ~~the~~ $min = j$ and a swap function is run to place the element in it's new position in the sorted sub array where it will be the biggest ellment in the sub array. meaning that $A[min]$ now represents a new element that is // the sorted array

The loop invariant holds.

Termination of program

At the termination of the inner loop, min indexes an element ~~less~~ larger than or equal to all elements in the array A [i ..... n] ~~always~~ as the array will be sorted an min will now index the largest elements making the loop invariant true.