EXPERIMENT 3

PRAKASH P. BISWAS

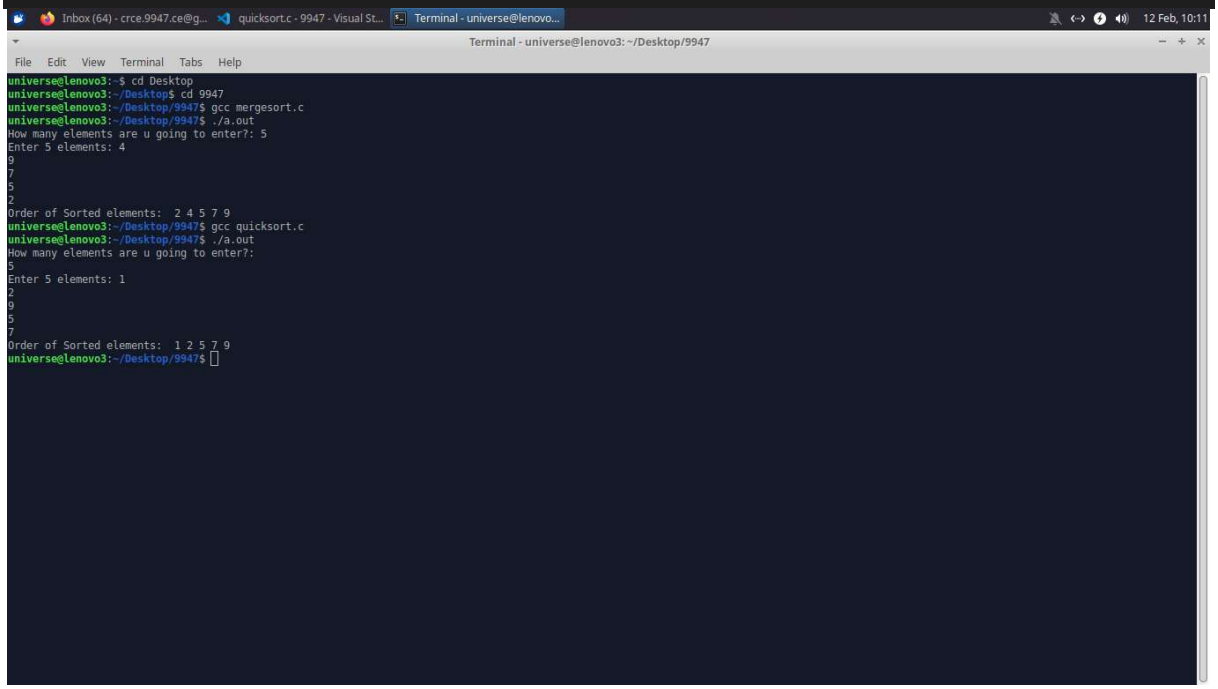
9947

COMPS-B

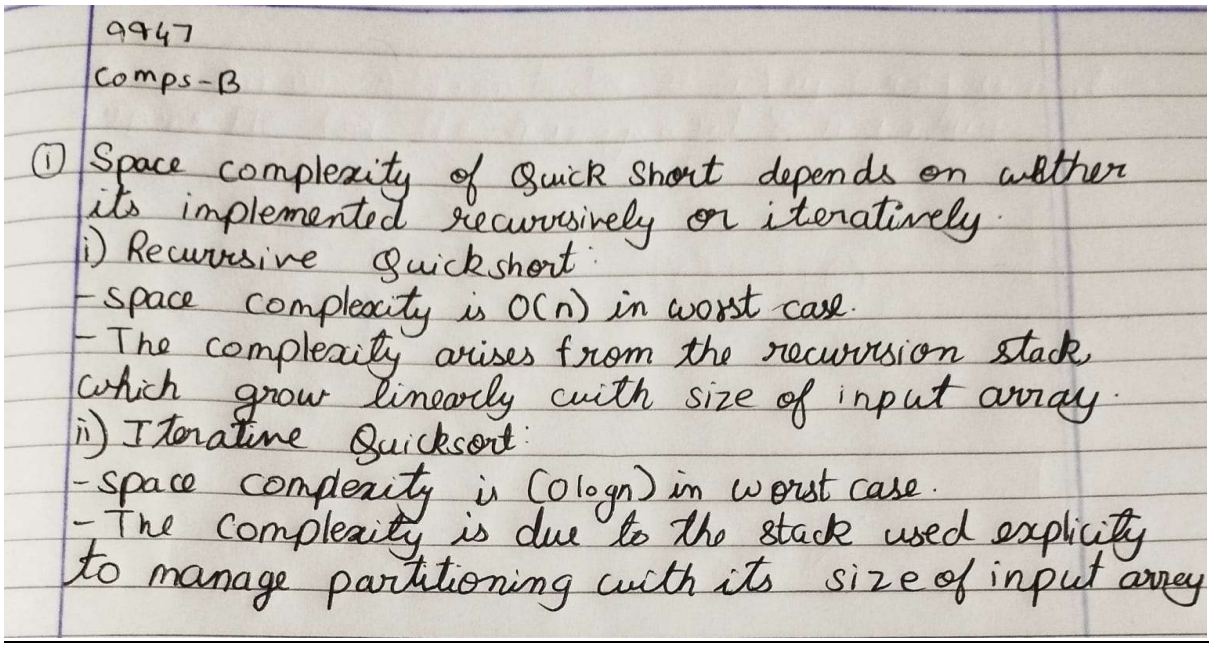
QUICK SORT

```
#include<stdio.h>
void quicksort(int no[25],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(no[i]<=no[pivot]&&i<last)
            i++;
         while(no[j]>no[pivot])
            j--;
         if(i<j){
            temp=no[i];
            no[i]=no[j];
            no[j]=temp;
         }
      }

      temp=no[pivot];
      no[pivot]=no[j];
      no[j]=temp;
      quicksort(no,first,j-1);
      quicksort(no,j+1,last);

   }
}

int main(){
   int i, count, no[25];

   printf("How many elements are u going to enter?: ");
   scanf("%d",&count);

   printf("Enter %d elements: ", count);
   for(i=0;i<count;i++)
      scanf("%d",&no[i]);
```

```
    quicksort(no,0,count-1);

    printf("Order of Sorted elements: ");
    for(i=0;i<count;i++)
       printf(" %d",no[i]);
    printf("\n");
    return 0;
}
```



POSTLAB:



9947
Comps-B

(1) Space complexity of Quick Short depends on wether its implemented recurrsively or iteratively.

i) Recurrsive Quickshort:
- space complexity is O(n) in worst case.
- The complexity arises from the recurrsion stack, which grow linearly with size of input array.

ii) Iterative Quicksort:
- space complexity is (O log n) in worst case.
- The complexity is due to the stack used explicity to manage partitioning with its size of input array