Question 1

```
#include <stdio.h>

int binary_search(int a[],int l,int h,int key){
    int mid=0;
    int count=0;
    int flag=3;
   while (l<h) {
         mid = (l+h)/2;

        if(a[mid]==key){
            count++;
            return mid;
        }
        else if(a[mid]>key){
            h=mid-1;
            flag=2;
        }
        else{
            l=mid+1;
        }
         }

    printf("no of comparisons are:%d\n",count+flag);

    return -1;
}

void linear_search(int a[],int key,int n){
    int flag=0;
```

```
    int count=0;
    for(int i=0;i<n;i++){
        if(a[i]==key){
            flag=1;
            break;
        }
       count++;
    }
    if(flag==1){
        printf("found\n");
    }
    else{
        printf(" not found");
    }
    printf("nunmberof comparisons are:%d\n",count);


}
int main()
{

    int a[5]={1,2,3,4,5};

    printf("%d\n",binary_search(a, 0, 4, 3)) ;
    linear_search(a, 2, 5);




return 0;
}
```

```
2
found
nunmberof comparisons are:1
Program ended with exit code: 0
```

Question 2

```c
#include <stdio.h>

struct faculty{
    char * fname[80];
    int faculty_id;
    int subject_code;
    char * class_name[80];
};

void bubble_sort(int a[],int n)
{
    int flag=0;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(a[j]>a[j+1]){
                int temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                flag=1;
            }
        }
        if(flag==1){
            break;
```

```
        }
    }
}

void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
    struct faculty f[5];
    int a[5];

    for(int i=0;i<5;i++){
        f[i].faculty_id=i+1;
    }

    for(int i=0;i<5;i++){
        a[i]=f[i].faculty_id;
    }
    bubble_sort(a, 5);

    printArray(a, 5);

    printf("TC in best case is O(n)\n");
    printf("TC in worst case and average case is O(n^2)\n");

return 0;
}
```

```
1 2 3 4 5
TC in best case is O(n)
TC in worst case and average case is O(n^2)
Program ended with exit code: 0
```

Question 3

Worst and average case

```c
#include <stdio.h>

void insertion_sort(int a[],int n){
    int count=1;
    for(int i=0;i<n;i++){
        int temp=a[i];

        int j=i-1;
        while (j>=0 && a[j]>temp) {
            a[j+1]=a[j];
            j--;
            count++;
        }
        a[j+1]=temp;
    }
    printf("%d\n",count);
    printf("time complexity is O(n^2)\n");
}
void selection_sort(int a[],int n){
    int count=1;

    for(int i=0;i<n-1;i++){
        int min=i;
```

```
        for(int j=i+1;j<n;j++){
            if(a[j]<a[min]){
                min=j;

            }
            if(min!=i){
                int temp=a[j];
                a[j]=a[min];
                a[min]=temp;
            }
            count++;
        }
    }
    printf("%d\n",count);
    printf("time complexity is O(n^2)\n");
}

int main()
{
    int a[5]={5,4,3,2,1};
    insertion_sort(a, 5);
    selection_sort(a, 5);
return 0;
}
```

```
11
time complexity is O(n^2)
11
time complexity is O(n^2)
Program ended with exit code: 0
```

Best case

```
#include <stdio.h>

void insertion_sort(int a[],int n){
```

```
    int count=1;
    for(int i=0;i<n;i++){
        int temp=a[i];

        int j=i-1;
        while (j>=0 && a[j]>temp) {
            a[j+1]=a[j];
            j--;
            count++;
        }
        a[j+1]=temp;
    }
    printf("%d\n",count);
    printf("time complexity is O(n)\n");
}
void selection_sort(int a[],int n){
    int count=1;
    int swaps=0;

    for(int i=0;i<n-1;i++){
        int min=i;
        for(int j=i+1;j<n;j++){
            if(a[j]<a[min]){
                min=j;

            }
            if(min!=i){
                int temp=a[j];
                a[j]=a[min];
                a[min]=temp;
                swaps++;
            }
            count++;
```

```
        }
    }
    printf("%d\n",count);
    printf("number of swaps are %d\n",swaps);
    printf("time complexity is O(n^2)\n");
}

int main()
{
    int a[5]={1,2,3,4,5};
    insertion_sort(a, 5);
    selection_sort(a, 5);
return 0;
}
```

```
1
time complexity is O(n)
11
number of swaps are 0
time complexity is O(n^2)
Program ended with exit code: 0
```

Question 4

```
#include <stdio.h>
```

```
int partition(int a[],int p ,int r)
{
    int x =a[r];
    int i=p-1;
    for(int j=p;j<r;j++){
        if(a[j]<=x){
            i++;
            int temp=a[j];
            a[j]=a[i];
            a[i]=temp;
        }
    }
    int temp=a[i+1];
    a[i+1]=a[r];
    a[r]=temp;

    return i+1;
}

void Quick_sort(int a[],int p,int r)
{
    if(p<r){
        int q=partition(a, p, r);
        Quick_sort(a,p,q-1);
        Quick_sort(a,q+1,r);
    }
}

int main()
{
    int a[5]={5,4,3,2,1};
    Quick_sort(a,0,4);
```

```
    for(int i=0;i<5;i++){
        printf("%d ",a[i]);
    }

return 0;
}
```

```
1 2 3 4 5 Program ended with exit code: 0
```

Line: 36 Col: 28

```
1 3 3 4 5 Program ended with exit code: 0
```

Line: 33 Col: 22

Question 5

```
#include <stdio.h>

struct student{
   char * name[80];
    int roll_no;
    int total_marks;
};

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];
```

```
for (i = 0; i < n1; i++)
    L[i] = arr[l + i];
for (j = 0; j < n2; j++)
    R[j] = arr[m + 1 + j];


i = 0;
j = 0;
k = l;
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    }
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
}


while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}


while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
```

```
    }
}


void mergeSort(int arr[], int l, int r)
{
    if (l < r) {

        int m = l + (r - l) / 2;


        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}


void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

int main()
{
   struct student s[10];
    for(int i=0;i<10;i++){
        s[i].roll_no=i;
    }// now we can intialise them
```

```
    int a[10];
    for(int i=0;i<10;i++){
        a[i]=s[i].roll_no;
    }

    mergeSort(a, 0, 10);
    printArray(a, 10);
    printf("Time complexity is nlogn in all 3 cases\n");

return 0;
}
```