```c
// Online C compiler to run C program online
#include <stdio.h>

void Bubblesort(int arr[], int n){
    int i, j;

    int temp;

    for(i=0; i<n-1; i++){
        for(j=0; j<n-i-1; j++){

            if(arr[j+1]<arr[j]){

                //swap
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
}

int main(){
    int i;
    int arr[5]={9,7,100,2,1};
    Bubblesort(arr,5);


    printf("Sorted array: \n");
    for(i=0; i<5; i++){
        printf("%d ", arr[i]);
    }
}

//_____________SELECTION
// Online C compiler to run C program online
#include <stdio.h>

void Selectionsort(int arr[], int n){

    int minPos, i, j,temp;

    for (i=0; i<n-1; i++){
        minPos=i;
        for (j=i+1; j<n;j++){
            if (arr[minPos]>arr[j]){
                minPos=j;
```

```
            }

        }

    //swap
    temp = arr[i];
    arr[i]=arr[minPos];
    arr[minPos]=temp;

    }
}

int main(){
    int i;
    int arr[5]={9,7,100,2,1};
    Selectionsort(arr,5);


    printf("Sorted: \n");
    for(i=0; i<5; i++){
        printf("%d ", arr[i]);
    }
}



//_______________INSERTION
// Online C compiler to run C program online
// Insertion sort in C

#include <stdio.h>



void insertionSort(int arr[], int size) {
    int curr, prev,i,j;

    for (i=1; i<size; i++){
        int curr=arr[i];
        int prev = i-1;

        while (prev>=0 && arr[prev]<curr){
            arr[prev+1]=arr[prev];
            prev--;
        }
```

```
        arr[prev+1]=curr;
    }
}

int main(){
    int i;
    int arr[5]={9,7,100,2,1};
    insertionSort(arr,5);


    printf("Sorted array: \n");
    for(i=0; i<5; i++){
        printf("%d ", arr[i]);
    }
}

//_____________QUICKSORT
#include <stdio.h>

void quickSort(int arr[], int first, int last){
    int i, j, pivot , temp;
    if(first<last)
    {
        pivot=first;
        i=first;
        j=last;
        while(i<j){
            while (arr[pivot]>=arr[i] && i<last){
                i++;}

            while(arr[pivot]<arr[j]){
                j--;
            }

            if(i<j){
                //swap
                temp = arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }

    temp=arr[pivot];
    arr[pivot]=arr[j];
    arr[j]=temp;
    quickSort(arr, first, j-1);
    quickSort(arr, j+1, last);
```

```
    }


}



int main(){
    int i;
    int arr[5]={9,7,-100,2,-1};
    quickSort(arr,0,5);


    printf("Sorted array: \n");
    for(i=0; i<5; i++){
        printf("%d ", arr[i]);
    }
}

//LINEAR WITH MULTIPLE

#include <stdio.h>

void linear(int a[], int size,int search){

    int i, flag=0, count=0;

    for (i=0; i<size; i++){
        if(a[i]==search){
            printf("Found at position %d\n",i+1);
            count++;
        }
    }

    if(count==0){
        printf("Not found");
        return;
    }

    printf("Found %d times", count);
}

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

```
}

//LINEAR WITH SINGLE

#include <stdio.h>

void linear(int a[], int size,int search){

    int i, flag=0, count=0;

    for (i=0; i<size; i++){
        if(a[i]==search){
            printf("Found at position %d\n",i+1);
            flag=1;
            return;
        }
    }

    if(flag==0){
        printf("Not found");
        return;
    }


}

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




//merge_____________________________
#include <stdio.h>
void merge(int a[],int low, int mid, int high){
    int i,j,k,b[20],r,x;

    i =low;
    j=mid+1;
    k=low;

    while(i<=mid && j<=high){
        if(a[i]<a[j]){
            b[k]=a[i];
            i++;
```

```
        }
        else{
            b[k]=a[j];
            j++;
        }

        k++;



    }

    if(i<=mid){
        for(r=i; r<=mid; r++){
            b[k]=a[r];
            k++;
        }
    }

    else{
        for(r=j; r<=high; r++){
            b[k]=a[r];
            k++;
        }
    }

    for(x=0; x<=high;x++){
        a[x]=b[x];

    }



}
void merge_sort(int a[],int low, int high){

    int mid ;
    if (low<high){
        mid = (low+high)/2;
        merge_sort( a,low,mid);
        merge_sort(a,mid+1, high);
        merge(a,low,mid, high);
    }
}
```

```
int main(){
    int i;
    int a[5]={9,7,100,2,1};
    merge_sort(a,0,4);


    printf("Sorted array: \n");
    for(i=0; i<5; i++){
        printf("%d ", a[i]);
    }
}

//BINARY RECURSIVE

// Online C compiler to run C program online
#include <stdio.h>

int bbinary(int a[], int start, int size, int search){
    int beg=start;
    int end=size-1;
   int mid;
   int flag=0;

   if(beg<=end)
   {
       mid = (beg+end) / 2;
       if(a[mid]==search){

           return mid;
       }

       if (a[mid]>search){
          return bbinary(a,start,mid-1,search);
       }

       return bbinary(a,mid+1, end, search);



   }



return -1;
 }
```

```
int main(){

    int arr[5]={56,98,100,500,2000};
   int pos = bbinary(a,0,5,100);

   if (pos==-1){
      printf("Not found");
   }
   else{
      printf("found at %d", pos);

   }

}

//normal binary
// Online C compiler to run C program online
#include <stdio.h>
void binary(int a[], int search, int size){
    int beg, end, mid, flag=0;
    beg=0;
    end=size-1;
    while(beg<=end){
        mid = (end+beg)/2;

        if(a[mid]==search){
            printf("Found at %d", mid+1);
            flag=1;
            return;
        }

        if(a[mid]>search){
            end = mid-1;
        }

        if(a[mid]<search){
            beg=mid+1;
        }
    }


    if(flag==0){
        printf("Not found");
    }
}
```

```
int main(){

    int arr[5]={1,2,3,4,5};
    binary(arr,6,5);
}
```