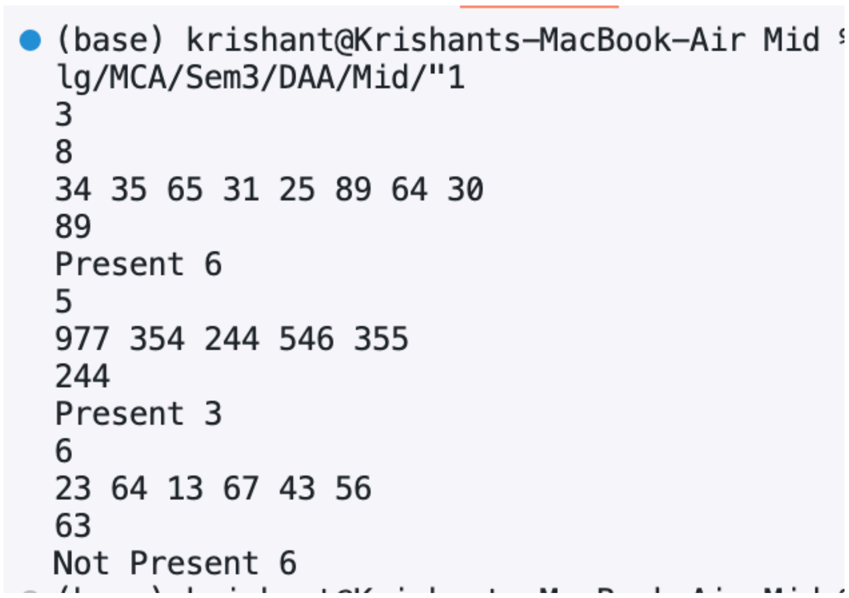
Q1-Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input).

Code:-

```
#include <stdio.h>
int main()
{
   int a;
   scanf("%d", &a);
   while (a)
   {
      int n;
      scanf("%d", &n);
      int arr[n];
      for (int i = 0; i < n; i++)
         scanf("%d", &arr[i]);

      int key;
      scanf("%d", &key);
      int flag = 0;
      int pos = 0;
      for (int i = 0; i < n; i++)
      {
         if (key == arr[i])
         {
            flag = 1;
            pos = i + 1;
            break;
         }
         else
            pos = i + 1;
      }
      if (flag)
         printf("Present %d \n", pos);
      else
         printf("Not Present %d \n", pos);
      a--;
   }
}
```

OUTPUT:-



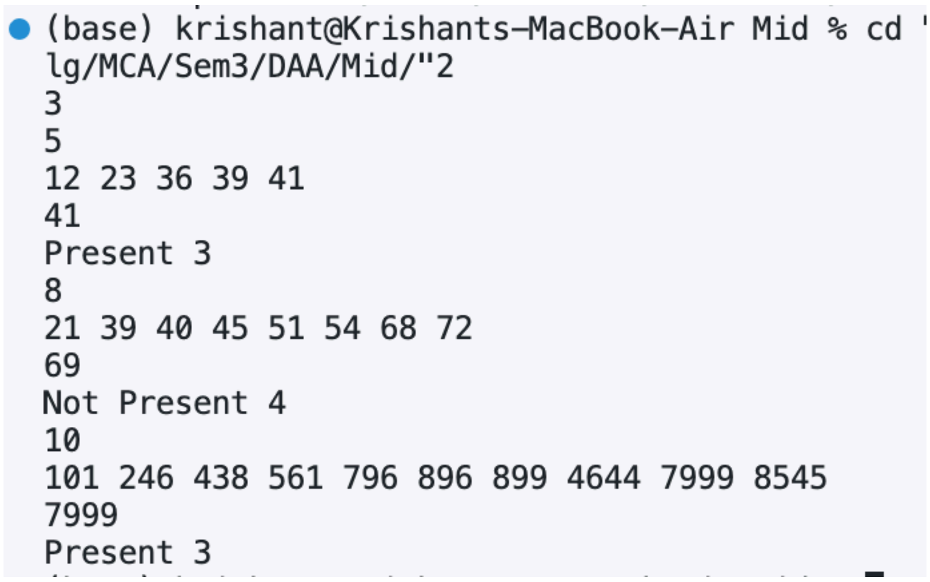
```
● (base) krishant@Krishants-MacBook-Air Mid 
  lg/MCA/Sem3/DAA/Mid/"1
  3
  8
  34 35 65 31 25 89 64 30
  89
  Present 6
  5
  977 354 244 546 355
  244
  Present 3
  6
  23 64 13 67 43 56
  63
  Not Present 6
```

Q2:-Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(logn), where n is the size of input).

Code:-

```
#include <stdio.h>
int binary_search(int arr[], int n, int key, int *ct)
{
   int low = 0, high = n - 1, mid;
   while (low <= high)
   {
      mid = low + (high - low) / 2;
      (*ct)++;
      if (arr[mid] == key)
         return mid;
      else if (arr[mid] < key)
         low = mid + 1;
      else
         high = mid - 1;
   }
   return -1;
}
int main()
{
   int a;
   scanf("%d", &a);
   while (a)
   {
      int n, key;
      scanf("%d", &n);
      int arr[n];
      for (int i = 0; i < n; i++)
         scanf("%d", &arr[i]);
      scanf("%d", &key);
      int ct = 0;
      int bs = binary_search(arr, n, key, &ct);
      if (bs != -1)
         printf("Present %d\n", ct);
      else
         printf("Not Present %d\n", ct);
      a--;
   }
   return 0;
}
```

OUTPUT:-



```
(base) krishant@Krishants-MacBook-Air Mid % cd '
lg/MCA/Sem3/DAA/Mid/"2
3
5
12 23 36 39 41
41
Present 3
8
21 39 40 45 51 54 68 72
69
Not Present 4
10
101 246 438 561 796 896 899 4644 7999 8545
7999
Present 3
```

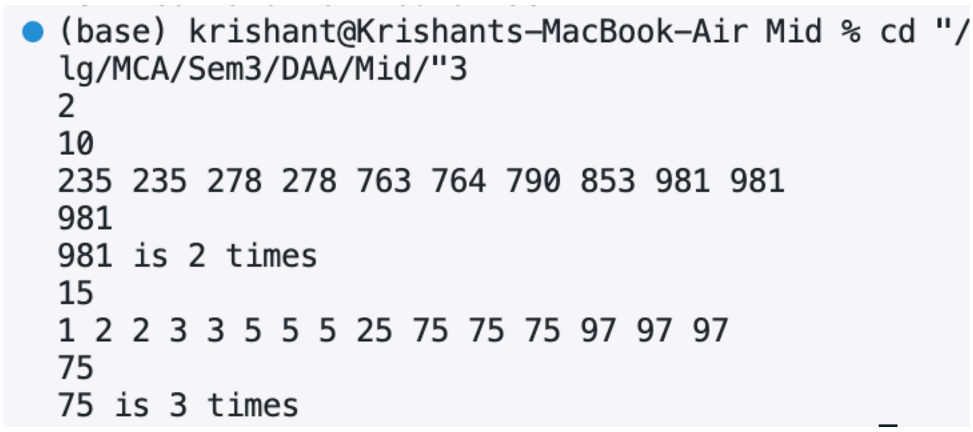
Q3:-Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

Code:-

```
#include <stdio.h>

int L(int arr[], int n, int key, int *ct)
{
    int low = 0, high = n - 1, pos = -1;

    while (low <= high)
    {
        int mid = low + (high - low) / 2;
        (*ct)++;

        if (arr[mid] == key)
        {
            pos = mid;
            high = mid - 1;
        }
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    return pos;
}

int R(int arr[], int n, int key, int *ct)
{
    int low = 0, high = n - 1, pos = -1;

    while (low <= high)
    {
        int mid = low + (high - low) / 2;
        (*ct)++;

        if (arr[mid] == key)
        {
            pos = mid;
            low = mid + 1;
        }
        else if (arr[mid] < key)
            low = mid + 1;
        else
```

```
            high = mid - 1;
    }

    return pos;
}

int main()
{
    int a;
    scanf("%d", &a);

    while (a)
    {
        int n, key;
        scanf("%d", &n);

        int arr[n];
        for (int i = 0; i < n; i++)
            scanf("%d", &arr[i]);

        scanf("%d", &key);

        int ct=0;
        int left = L(arr, n, key, &ct);
        int right = R(arr, n, key, &ct);

        if (left != -1)
        printf("%d is %d times\n",key, right - left + 1);
        else
            printf("Not Present\n");
        a--;
    }

    return 0;
}
```

OUTPUT:-



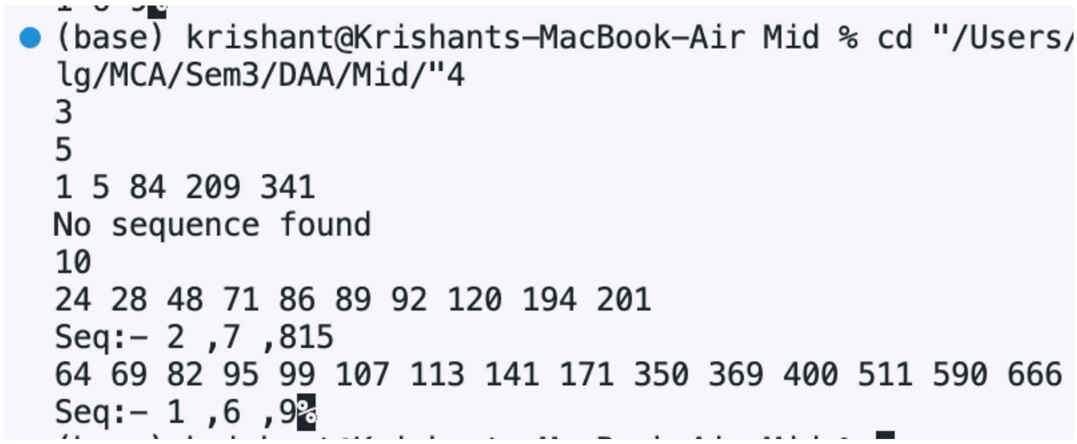
```
● (base) krishant@Krishants-MacBook-Air Mid % cd "/
  lg/MCA/Sem3/DAA/Mid/"3
  2
  10
  235 235 278 278 763 764 790 853 981 981
  981
  981 is 2 times
  15
  1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
  75
  75 is 3 times
```

Q4:-Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

Code:-
```
#include <stdio.h>
int binary_search(int arr[], int n, int key)
{
   int low = 0, high = n - 1, mid;
   while (low <= high)
   {
      mid = low + (high - low) / 2;

      if (arr[mid] == key)
         return mid;
      else if (arr[mid] < key)
         low = mid + 1;
      else
         high = mid - 1;
   }
   return -1;
}

int main()
{
   int a;
   scanf("%d", &a);

   while (a)
   {
      int n, key;
      scanf("%d", &n);

      int arr[n];
      for (int i = 0; i < n; i++)
         scanf("%d", &arr[i]);

      int flag = 0;

      for (int i = 0; i < n; i++)
      {
         for (int j = i + 1; j < n; j++)
         {
            int sum = arr[i] + arr[j];
            int bs = binary_search(arr, n, sum);
            if (bs != -1)
            {
               flag = 1;
```

```
                printf("Seq:- %d ,%d ,%d", i + 1, j + 1, bs + 1);
                break;
            }
            else
                flag = 0;
        }
        if (flag)
            break;
    }
    if (!flag)
        printf("No sequence found\n");
    a--;
  }

  return 0;
}
```

OUTPUT:-



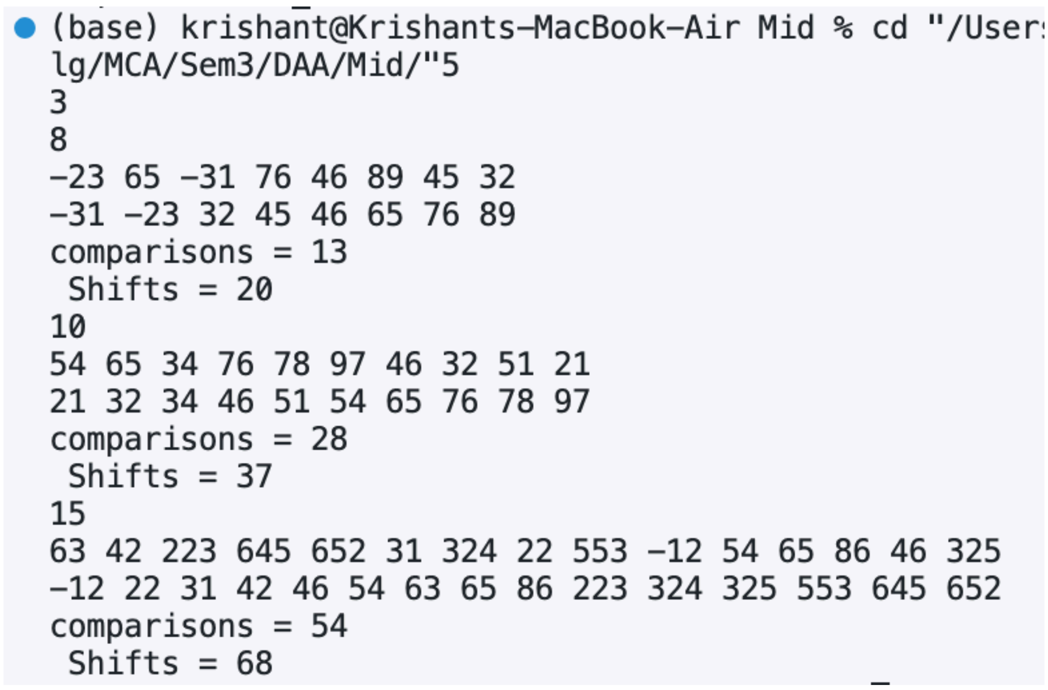
```
● (base) krishant@Krishants-MacBook-Air Mid % cd "/Users/
  lg/MCA/Sem3/DAA/Mid/"4
  3
  5
  1 5 84 209 341
  No sequence found
  10
  24 28 48 71 86 89 92 120 194 201
  Seq:- 2 ,7 ,815
  64 69 82 95 99 107 113 141 171 350 369 400 511 590 666
  Seq:- 1 ,6 ,9%
```

Q5:-Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts - total number of times the array elements are shifted from their place) required for sorting the array.

Code:-

```
#include <stdio.h>
void in(int arr[], int n, int *cmp, int *sft)
{
   for (int i = 1; i < n; i++)
   {
      int key = arr[i];
      int j = i - 1;
      while (j >= 0 && key < arr[j])
      {
         (*cmp)++;
         (*sft)++;
         arr[j + 1] = arr[j];
         j--;
      }
      (*sft)++;
      arr[j + 1] = key;
   }
}
int main()
{
   int a;
   scanf("%d", &a);
   while (a)
   {
      int n, key;
      scanf("%d", &n);
      int arr[n];
      for (int i = 0; i < n; i++)
         scanf("%d", &arr[i]);
      int cmp = 0, sft = 0;
      in(arr, n, &cmp, &sft);
      for (int i = 0; i < n; i++)
         printf("%d ", arr[i]);
      printf("\ncomparisons = %d\n Shifts = %d\n", cmp, sft);
      a--;
   }
   return 0;
}
```

OUTPUT:-



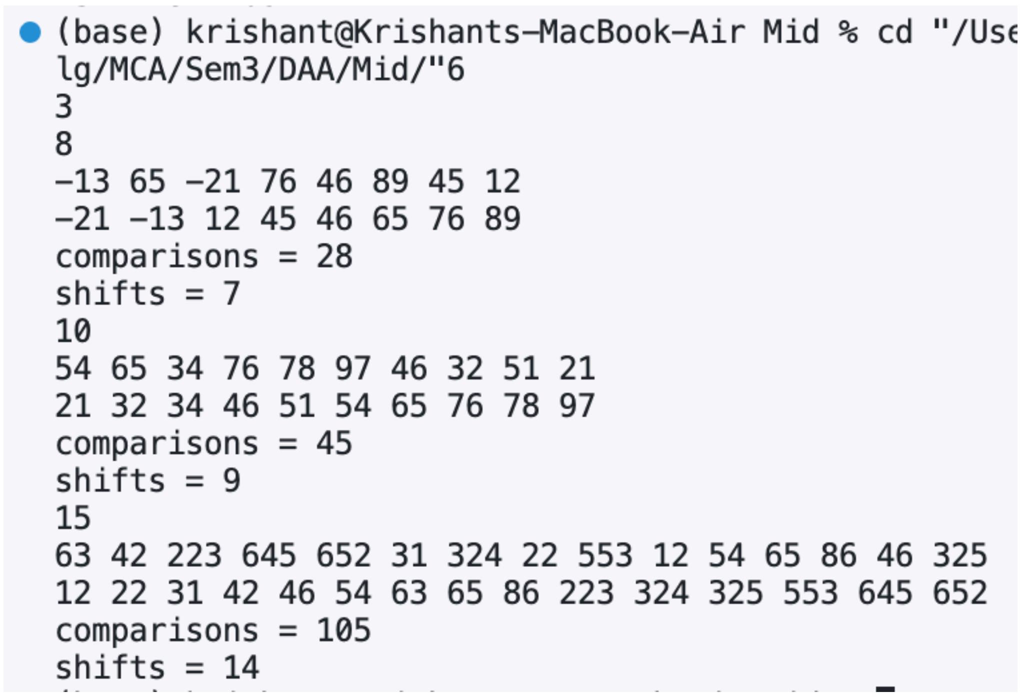
```
● (base) krishant@Krishants-MacBook-Air Mid % cd "/User
  lg/MCA/Sem3/DAA/Mid/"5
  3
  8
  -23 65 -31 76 46 89 45 32
  -31 -23 32 45 46 65 76 89
  comparisons = 13
   Shifts = 20
  10
  54 65 34 76 78 97 46 32 51 21
  21 32 34 46 51 54 65 76 78 97
  comparisons = 28
   Shifts = 37
  15
  63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325
  -12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
  comparisons = 54
   Shifts = 68
```

Q6:-Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

Code:-

```
#include <stdio.h>

void selection(int arr[], int n, int *cmp, int *sft)
{
    for (int i = 0; i < n - 1; i++)
    {
        int mn = i;
        for (int j = i + 1; j < n; j++)
        {
            (*cmp)++;
            if (arr[j] < arr[mn])
                mn = j;
        }
        (*sft)++;
        int temp = arr[i];
        arr[i] = arr[mn];
        arr[mn] = temp;
    }
}
int main()
{
    int a;
    scanf("%d", &a);
    while (a)
    {
        int n, key;
        scanf("%d", &n);
        int arr[n];
        for (int i = 0; i < n; i++)
            scanf("%d", &arr[i]);
        int cmp = 0, sft = 0;
        selection(arr, n, &cmp, &sft);
        for (int i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\ncomparisons = %d\nshifts = %d\n", cmp, sft);
        a--;
    }
    return 0;
}
```

OUTPUT:-



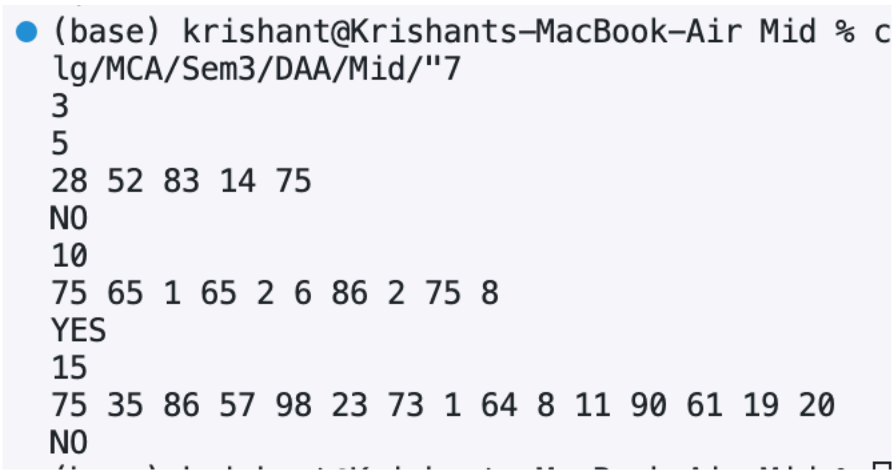
```
● (base) krishant@Krishants-MacBook-Air Mid % cd "/Use
  lg/MCA/Sem3/DAA/Mid/"6
  3
  8
  -13 65 -21 76 46 89 45 12
  -21 -13 12 45 46 65 76 89
  comparisons = 28
  shifts = 7
  10
  54 65 34 76 78 97 46 32 51 21
  21 32 34 46 51 54 65 76 78 97
  comparisons = 45
  shifts = 9
  15
  63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
  12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
  comparisons = 105
  shifts = 14
```

Q7:- Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

Code:-
```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int left, int mid, int right) {
   int i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;
   int L[n1], R[n2];
   for (i = 0; i < n1; i++)
      L[i] = arr[left + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[mid + 1 + j];
   i = 0;
   j = 0;
   k = left;
   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 left, int right) {
   if (left < right) {
      int mid = left + (right - left) / 2;
      mergeSort(arr, left, mid);
      mergeSort(arr, mid + 1, right);
      merge(arr, left, mid, right);
   }
}
```

```
int hasDuplicates(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        if (arr[i] == arr[i + 1])
            return 1;
    }
    return 0;
}
int main() {
    int a;
    scanf("%d", &a);
    while (a--) {
        int n;
        scanf("%d", &n);
        int arr[n];
        for (int i = 0; i < n; i++)
            scanf("%d", &arr[i]);
        mergeSort(arr, 0, n - 1);
        if (hasDuplicates(arr, n))
            printf("YES\n");
        else
            printf("NO\n");
    }

    return 0;
}
```

OUTPUT:-



```
● (base) krishant@Krishants-MacBook-Air Mid % c
  lg/MCA/Sem3/DAA/Mid/"7
  3
  5
  28 52 83 14 75
  NO
  10
  75 65 1 65 2 6 86 2 75 8
  YES
  15
  75 35 86 57 98 23 73 1 64 8 11 90 61 19 20
  NO
```

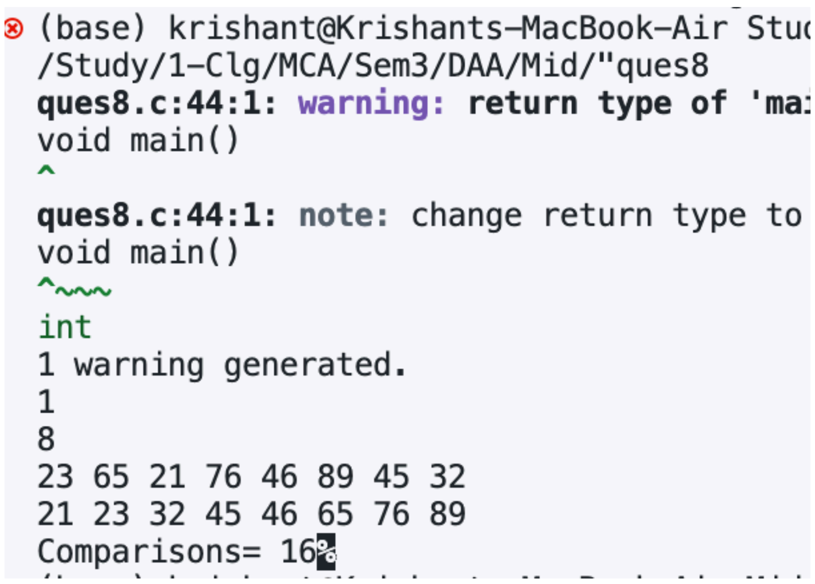
Q8:-Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.

Code:-

```
#include <stdio.h>
static int comp = 0;
void merge(int arr[], int l, int mid, int r)
{
   int n1 = mid - l + 1;
   int n2 = r - mid;
   int left[n1], right[n2];
   for (int i = 0; i < n1; i++)
      left[i] = arr[l + i];
   for (int i = 0; i < n2; i++)
      right[i] = arr[mid + 1 + i];
   int i = 0, j = 0, k = l;
   while (i < n1  && j < n2)
   {
      if (left[i] < right[j])
      {
         arr[k] = left[i];
         i++;
      }
      else
      {
         arr[k] = right[j];
         j++;
      }
      k++;
      comp++;
   }
   while (i < n1)
      arr[k++] = left[i++];
   while (j < n2)
      arr[k++] = right[j++];
}

void mergeSort(int arr[], int l, int r)
{
   if (l >= r)
      return;
   int mid = (l + r) / 2;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid + 1, r);
   merge(arr, l, mid, r);
}
```

```
void main()
{
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        int size;
        scanf("%d", &size);
        int arr[size];
        for (int j = 0; j < size; j++)
            scanf("%d", &arr[j]);
        int flag = 0;
        mergeSort(arr, 0, size - 1);
        for (int j = 0; j < size; j++)
            printf("%d ", arr[j]);
        printf("\nComparisons= %d", comp);
    }
}
```

OUTPUT:-



```
(base) krishant@Krishants-MacBook-Air Stu
/Study/1-Clg/MCA/Sem3/DAA/Mid/"ques8
ques8.c:44:1: warning: return type of 'ma
void main()
^
ques8.c:44:1: note: change return type to
void main()
^~~~
int
1 warning generated.
1
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparisons= 16%
```

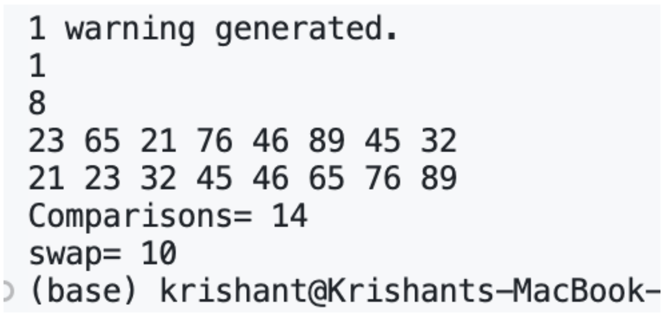
Q9:-Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another subarray holds values greater than the pivot element. Pivot element should be selected randomly from the array.Your program should also find number of comparisons and swaps required for sorting the array.

Code:-

```
#include<stdio.h>
static int comp=0;
static int sp=0;
void swap(int* a, int* b)
{
   int t = *a;
   *a = *b;
   *b = t;
}
int partition(int arr[], int low, int high)
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high - 1; j++) {
      if (arr[j] < pivot) {
         i++;
         swap(&arr[i], &arr[j]);
         sp++;
      }
      comp++;
   }
   sp++;
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
   if (low < high) {
      int pi = partition(arr, low, high);
      quickSort(arr, low, pi - 1);
      quickSort(arr, pi + 1, high);
   }
}
void main()
{
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      int size;
```

```
        scanf("%d",&size);
        int arr[size];
        for(int j=0;j<size;j++)
        {
           scanf("%d",&arr[j]);
        }
        quickSort(arr,0,size-1);
        for(int j=0;j<size;j++)
        {
           printf("%d ",arr[j]);
        }
        printf("\nComparisons= %d",comp);
        printf("\nswap= %d\n",sp);
        sp=0;comp=0;
    }
}
```

Code:-



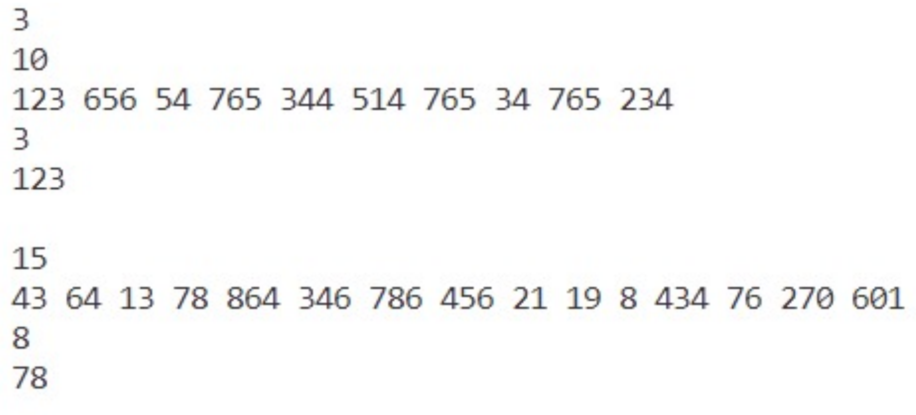
```
1 warning generated.
1
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparisons= 14
swap= 10
(base) krishant@Krishants-MacBook-
```

Q10:-Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))

Code:-

```
#include<stdio.h>
void main()
{
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      int size;
      scanf("%d",&size);
      int arr[size];
      int max=0;
      for(int j=0;j<size;j++)        {
         scanf("%d",&arr[j]);
         if(max<arr[j])
            max=arr[j];        }
      int num;
      scanf("%d",&num);
      int countSort[max+1];
      for(int j=0;j<size;j++)
         countSort[arr[j]]++;
      for(int j=0;j<=max;j++)        {
         num-=countSort[j];
         if(num<=0)           {
            printf("%d\n",j);
            break;
         }        }
}}
```

OUTPUT:-



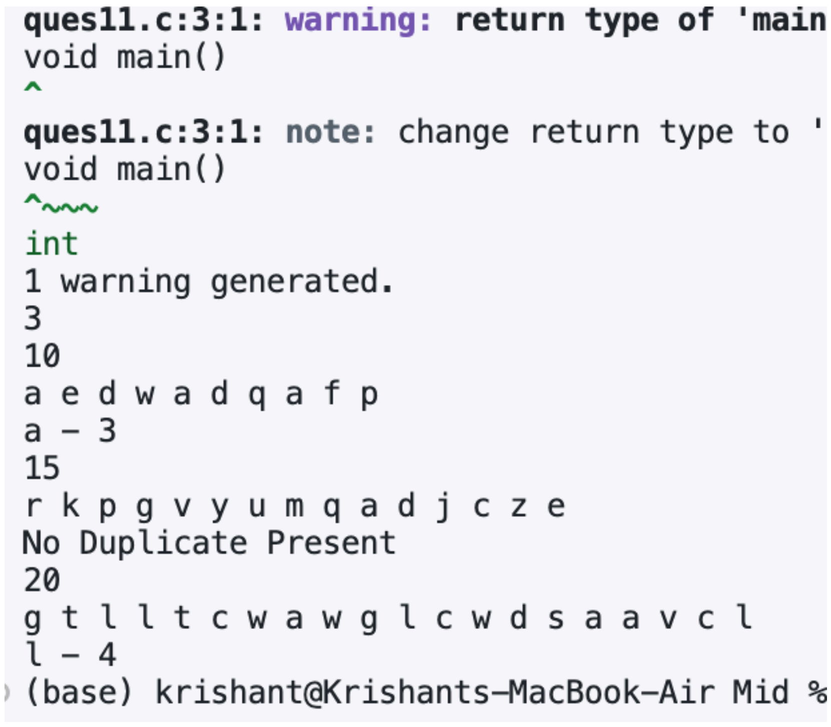
```
3
10
123 656 54 765 344 514 765 34 765 234
3
123

15
43 64 13 78 864 346 786 456 21 19 8 434 76 270 601
8
78
```

Q10:-Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and
implement it using a program to find which alphabet has maximum number of occurrences and
print it. (Time Complexity = O(n)) (Hint: Use counting sort)

Code:-

```
#include<stdio.h>

void main()
{
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      int size;
      scanf("%d",&size);
      char arr[size];
      for(int j=0;j<size;j++)
      {
         scanf(" %c",&arr[j]);
      }
      int alph[26]={0};
      for(int j=0;j<size;j++)
      {
         alph[arr[j]-'a']++;
      }
      int maxCount=0;
      int maxChar;
      for(int j=0;j<26;j++)
      {
         if(alph[j]>maxCount)
         {
            maxCount=alph[j];
            maxChar=j+'a';
         }
      }
      if(maxCount==1)
      {
         printf("No Duplicate Present");
      }
      else
      {
         printf("%c - %d\n",maxChar,maxCount);
      }
   }
}
```

OUTPUT:-



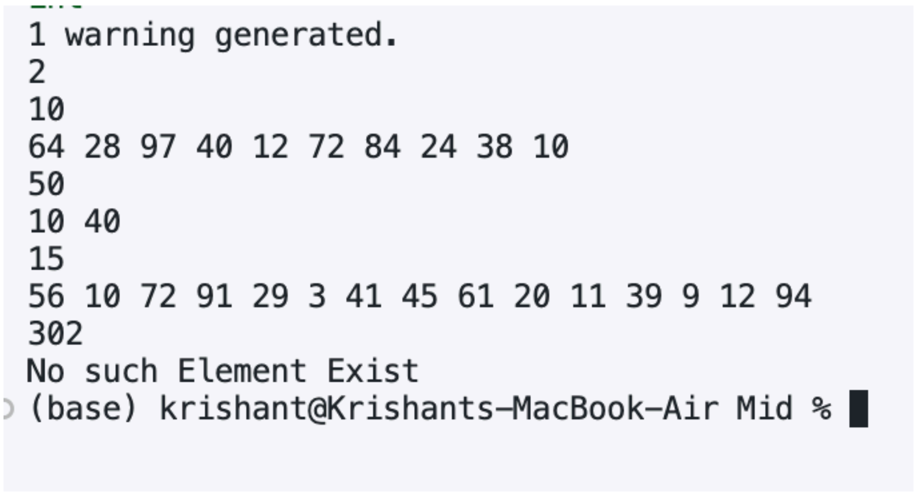
```
ques11.c:3:1: warning: return type of 'main
void main()
^
ques11.c:3:1: note: change return type to '
void main()
^~~~
int
1 warning generated.
3
10
a e d w a d q a f p
a - 3
15
r k p g v y u m q a d j c z e
No Duplicate Present
20
g t l l t c w a w g l c w d s a a v c l
l - 4
(base) krishant@Krishants-MacBook-Air Mid %
```

Q11:-Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))

Code:-
```
#include<stdio.h>
void swap(int* a, int* b)
{
   int t = *a;
   *a = *b;
   *b = t;
}
int partition(int arr[], int low, int high)
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high - 1; j++) {
      if (arr[j] < pivot) {
         i++;
         swap(&arr[i], &arr[j]);
      }
   }
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}
void quickSort(int arr[], int low, int high)
{
   if (low < high) {
      int pi = partition(arr, low, high);
      quickSort(arr, low, pi - 1);
      quickSort(arr, pi + 1, high);
   }
}
void main()
{
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      int size;
      scanf("%d",&size);
      int arr[size];
      for(int j=0;j<size;j++)
      {
         scanf("%d",&arr[j]);
      }
      int key;
      scanf("%d",&key);
```

```
        quickSort(arr,0,size-1);
        int l=0,r=size-1;
        while(l<r)
        {
            if(arr[l]+arr[r]==key)
            {
                printf("%d %d\n",arr[l],arr[r]);
                break;
            }
            else if(arr[l]+arr[r]<key)
                l++;
            else
                r--;
        }
        if(l>=r)
        {
            printf("No such Element Exist\n");
        }
    }
}
```

OUTPUT:-



```
1 warning generated.
2
10
64 28 97 40 12 72 84 24 38 10
50
10 40
15
56 10 72 91 29 3 41 45 61 20 11 39 9 12 94
302
No such Element Exist
(base) krishant@Krishants-MacBook-Air Mid %
```