DSA
Assignment-6

Tummapudi Girishma
AP19110010017
CSE-7

1.
```
#include <stdlib.h>
#include <stdio.h>
int comparator (const void *p1, const void *p2)
{
return (*(int*)p2 - *(int*)p1);
}
int binary serch (int arr[], int size, int serch){
    int beg=0, end=size-1, mid;
    while (beg<=end){
        mid=(beg+end)/2
        if (arr[mid]==serch){
            return mid;
        }
        else if (arr[mid]<serch){
            end=mid-1;
        }else beg=mid+1;
}
return -1
int main()
{
    int arr[100], size, serch, i, pos=-1, loc1, loc2;
    printf("\n Enter the size of the array(max100)");
    scanf("%d", &size)
    printf("\n Enter elements in array \n");
    for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
    }
qsort(arr, size, sizeof(int), comporator);
printf("\n The sorted array is: \n");
    for(i=0; i<size; i++)
```

```
printf("%d", arr[i]);
}
printf("\n Enter search element");
scanf("%d", &search);
pos = binary search (arr, size, search);
if (pos == -1) printf("Not found");
else printf("\n the %d search element is
found at index %d \n", serch, pos);
printf("Enter two indexes \n");
scanf("%d %d", &loc1, &loc2);
printf("sum is %d \n", arr[loc1] + arr[loc2]);
printf("product is %d \n", arr[loc1] * arr[loc2]
```

2.
```
#include <stdlib.h>
#include <stdio.h>
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)
{
```

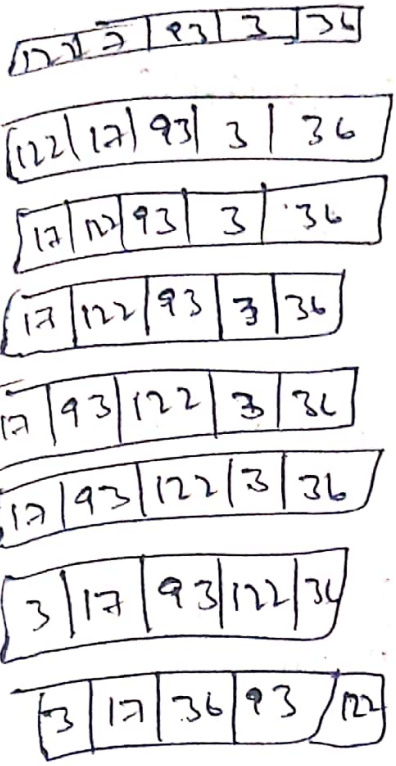
```
if (L[i] <= R[j])
{
arr[k] = L[i];
i++
}
else
{
arr[k] = R[j];
j++
}k++;
}
while (i<n1)
{arr[k] = L[i];
i++;
k++;
}
while (j<n2)
{
arr[k] = R[j];
j++
k++
}
}
void merge sort(int arr[], int l, int r)
{ if (l<r)
{int m = l+(r-l)/2;
merge sort(arr, l, m);
merge sort(arr, m+1, r);
merge(arr, l, m, r);
}
}
int main()
{
int a[100], n, k;
```

```
printf("Enter the number of elements");
scanf("%d", &n);
for(int i=0;i<n;i++){
printf("Enter next element");
scanf("%d", &a[i]);

}
mergeSort(a,0,n-1);
printf("Sorted Array:");
for(int i=0;i<n;i++)
printf("%d", a[i]);
printf("\n Enter k value to find the product
of kth element from first and last");
scanf("%d",&k);
printf("The product is: %d", a[k-1]*a[n-k]);
return 0;
}
```

3. Insertion Sort:

1. If the element in first one, it is already sorted
2. Move to text element
3. Compare the coment element with all elements in sorted array
4. If the element in sorted array is small than coment element, iterate to the next element otherwise shift all the greater element in array lay one position toward right
5. Insert the value at the cret position
6. Repeat until the complete list is sorted



→ 122, 17, 93, 3, 36

for i = 1 (2nd element to 36
i=1 since 7 is smaller than 122, move 122 and insert 17 before 122.

→ 17, 122, 93, 3, 36

i = 2 since 93 is smaller than 122, move 122 and insert 93 before 123

→ 7, 93, 122, 3, 36

i = 3 & 3 will to the begin and all other element from 17 to 122 will move one portion ahead of their current portion

→ 3, 17, 93, 122, 36

i = 4 36 will move to position after 17 and elements form 93 to 122 will move one position ahead of their current position

→ 3, 17, 36, 93, 122.

Selection Sort: Consider array [10, 5, 2, 1]
the 1st element is 10 in next part we must find the smallest number from remaining array the smallest number from 5, 2 & 1 So, we replace 10 by 1

The new array [1, 5, 2, 10] Again the process repeats

Finally we get the sorted array as [1, 2, 5, 10]

→ Set min to first location
→ Search minimum element in array
→ Swap the first location with minimum value with array
→ assign the second element as main.
→ Repeat the process until we get a sorted array.

```
#include <stdio.h>
void displayAltSumPro(int arr[], int size){
    int i, sum=0, product=1;
    printf("Alternate elements \n")
    for(i=0; i<size; i++){
        if(i%2!=0){
            product *= arr[i];
        }else{
            sum+=arr[i];
            printf("%d ", arr[i]);
        }
    }
    printf("\n Sum of odd elements = %d\n", sum)
    printf("\n product of the even elements
                                = %d\n", pod
}
void divM(int arr[], int size){
    int i=0, m;
    printf("Enter the m\n");
    scanf("%d", &m);
    printf("elements divisible by %d \n", m);
    for(i=0; i<size, i++){
        if(arr[i]%m==0)
            printf("%d ", arr[i]);
    }
}
void bubbleSort(int arr[], int size)
{ int i, j, temp;
for(i=0; i<size-1; i++)
  for(j=0; j<size-i-1; j++)
    if(arr[j]>arr[j+1]){
```

```
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp
        }
    }
    displayAltSumPro(arr, size);
    divAll(arr, size);
}
int main()
{ int arr[100], size, i;
  printf("\n Enter elements in array \n");
  for (i=0; i<size; i++) {
        scanf("%d", &arr[i]);
  }
  bubbleSort(arr, size-1);
  return 0;
```

5
```
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
  if (r >= l) {
  int mid = l + (r-l)/2;
  if (arr[mid] == x)
  return mid;
  return binarySearch(arr, l, mid-1, x);
  return binarySearch(arr, mid+1, r, x);
  }
  return -1;
}
int main(void)
{
  int a[100], n, r, x;
  printf("\n Enter the elements in ascending order
                                    only \n");
```

```
printf("Enter the number of elements:");
scanf("%d", &n);
for(int i=0; i<n; i++){
printf("Enter next element:");
scanf("%d", &a[i]);
}
printf("Enter the element to be serched:");
scanf("%d", &x);
int result = binary_search(a, 0, n-1, x);
(result = -1)? printf("Elements are not presented
                in array\n"):printf("Ele
-ents is present at index %d\n", result);
```

}.

Output:

```
// Enter the elements in accending order only!
Enter the number of elements: 5
Enter next element:
1
2
3
6
5
Enter the element to be serched: 6
Elements is present at index 4.
```

Output for 1,2,3

```
① Enter the size of the array (max 100)5
Enter elements in array
5 2 3 6 7
```

The sorted array is:
7 6 5 3 2
Enter search elements 2
the 2 serch element is found at index 4
Enter two indexes
2 3
sum is 8
product is 15.

(3) Enter the number of elements
a[0] = 1
a[1] = 6
a[2] = 1
a[3] = 54
a[4] = 2
Enter K value to find the product of K^{th} element from first and last : 3
the product is : 2

(4) Enter the size of the array (max 100) 5
Enter elements in array
9 4 5 2 8
Alternate elements
2 5
Sum of the odd elements = 7
product of the even elements = 14
Enter the m
3
elements divisible by 3

9.