Assignment-6

Name:- Asritha. Meka
ROLL: AP19110010224
CSE-G

① Take the elements from the user and sort them in decending order and do ⓐ using binary search ⓑ ASK the user to enter any 2 locations-

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

int comparator (const void *P1, const void *P2)
{
return (*(int*)P2 - *(int*)P1);
}
int binarySeach(int arr[], int size, int Search){
    int beg=0, end=size-1, mid;
    while (beg<=end){
        mid=(beg+end)/2;
        If (arr[mid]==Search){
            return mid;
        }
        else if (arr(mid]<search){
            end=mid-1;
        }
        return-1;
    }
    int main()
{
    int arr[100], size, Search, i, Pos=-1, loc1,
                                    loc2;
    printf("\nEnter the size of the array
     (max 100)");
```

```
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), comparator);
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 = binarysearch(arr, size, search);
if (pos==-1) printf("Not found \n");
else printf("\n the %d search element
    is found at index %d\n", search, 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]);
}
```

output

Enter the size of the array [max 100] 5.
Enter elements in array : 5 2 3 6 9
The sorted array is: 9 6 5 3 2.
enter search element 2.
the 2 search element is found at index 4
Enter two indexes. 1 2
sum is 11
product is 30

---

(2) Sorted the array using Merge sort where elements are taken from the user and find the product of kth elements.

```
#include <stdio.h>
#define ms 100
int a[ms];
void merge(int l1, int u1, int l2, int u2)
{
  int i, j, k, temp[ms];
  k=0;
  i=l1;
  j=l2;
  while((i<=u1)&&(j<=u2)){
    if(a[i]<a[j]){
      temp[k]=a[i];i++;k++;
    }
    else{
      temp[k]=a[j];j++;k++;
```

```
    }
    else {
      temp[k] = a[j]; j++; k++;
    }
  }
  while (i<=u1) {
    temp[k] = a[i]; i++; k++;
  }
  while (j<=u2) {
    temp[k] = a[j]; j++; k++;
  }
  while (j<=u) {
    temp[k] = a[j]; j++; k++;
  }
  for (i=l1, k=0; i<=u2; i++, k++) {
    a[i] = temp[k];
  }
}
void mergesort(int lb, int ub) {
  if (lb<ub)
  {
    int mid = (ub+lb)/2;
    mergesort(lb, mid);
    mergesort(mid+1, ub);
    merge(lb, mid, mid+1, ub);
  }
}
```

```
int main(){
    int i,n, product = 1, k;
printf("\n Enter the size of the array
            max(100) ");
 scanf("%d", &n);
 for(i=0; i<n; i++){
   printf("a[%d] \t =", i);
   scanf("%d", &a[i]);
 }
 mergesort(0, n-1);
 printf("enter k\n");
 scanf("%d", &k);
 for(i=0; i<k; i++){
     product *= a[i];
 }
   printf("\n the product till the k.
     the element is %d \n", product);
 return 0;
 }
```

output

```
Enter the size of the array 5
a[0] = 1
a[1] = 5
a[2] = 1
a[3] = 55
a[4] = 4
Enter k
3
```

the product till the k th element is 4

(5) Dicuss insertion sort and selection sort with examples.

insertion sort:

An array A with n elements A[1], A[2], --- A[N]. is in memory. the insertion sort algorithm scans A form A[1] to A[N], insertion each elem ent A[K] into its proper position in the previous sorted sub array A[1], A[2], ---- A[K-1].

Examples

array intial: 15, 19, 12, 21, 9.

pass1 : 15, 19, 12, 21, 9

p2 : 12, 15, 19, 21, 9

p3 : 12, 15, 19, 21, 9

p4 : 9, 13, 15, 19, 21 sorted.

Code:

(1) A[10] = minium integer value

(2) Repeat steps 3 trough 8 for K = 1, 2, 3, ---, N-1.

:

(3) temp = A[K]

(4) ptr = K-1

(5) Repeat steps 6 to 7 while temp < A[ptr]

{

⑥ A(ptr+1) = A[ptr]

⑦ ptr = ptr - 1

}

⑧ A[ptr+1] = temp

}

⑨ END

time complexity.

best: O(n) average O(n^2) worst O(n^2)

Space complexity: O(1)

Selection Sort

the basic idea of selection sort is repeatedly select the smallest key in the unsorted array.

ex:- 15, 6, 13, 3.

P① :- 2 3 15, 6, 13, 3.

P② :- 2, 3 15, 6, 13

P③ :- 2, 3, 6 15, 13

P④ :- 2, 3, 6, 13 15

P⑤ :- 2, 3, 6, 13, 15.

pseudo code.

① small = np[4]

② for i < 2 to u to {

(3) 3. small = AR[i], POS]

4. for 5 = 1 to do {

5. if AR[i] < small then {

small = AR[i], POS[i]

6. }

7. J = J + 1

8. temp = AR[i], AR(i) = small

}

---

(4) sorting the array using bubble sort where
(i) in alternate order.
(ii) sum of elements in odd positions & product of elements.
(iii) elements which are divisible by m.

```
# include <stdio.h>
void display Alt sum pro (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 the odd elements =
                %d\n", sum);
printf("\n product of the even elements
            = %d\n", product);
}
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;
        }
    }
    displayAltSumProd(arr, size);
    divM(arr, size);
}
int main()
{
    int arr[100], size, i;
    printf("\n Enter the size of the
            array (max 100)");
    scanf("%d", &size);
    printf("\n Enter elements in array\n");
    for (i=0; i<size; i++) {
        scanf("%d", &arr[i]);
    }
    bubbleSort(arr, size-1);
    return 0;
```

output:-

```
Enter size (max 100) 5
enter elements: 10 3 4 1 7
  Alternate elements | 4
Sum of odd elements = 5
  product = 4
enter the m
  2
element divisible by 2
  4
```

⑤ Write a recursive program to implement binary search -?

Coding:-

```
#include <stdio.h>
int binary search (int arr[], int beg, int end,
  int search) {
   int mid;
    if (beg <= end) {
      mid = (beg + end) / 2;
       if (arr[mid] == search) return mid;
      if (arr[mid] > search) {
            return binary search (arr, beg,
                 mid - 1, search)
      }
      return binary search (arr, mid+1,
          end, search);
    }
    return -1;
}
int main()
{
   int arr[100], size, search, i, pos;
   printf ("\n Enter the size of array
                (max 100)");
   scanf ("%d", &size);
   printf ("\n Enter sorted elements in
      array \n");
```

```
for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
}
printf("\n Enter Search element");
scanf("%d", &search);
pos = binarySearch(arr, 0, size-1, search);
if (pos==-1) printf("Not found\n");
else printf("\n the %d Search element
        is found at index %d\n", search, pos);
  return 0;
}
```

output :

```
enter the size of array[max 100] 5
enter sorted elements
1
2
3
4
5
enter Search element 2
the 2 search element is found at
        index 1
```