Assignment-6

Searching & Sorting in C.

Pragya Gupta
AP19110010532.
CSE-G.

```
① #include <stdio.h>
  #include <stdlib.h>

 int comparator(const void *p1,const void *p2){
  return (*(int*)p2 - *(int*)p1);
 }

 int Binary search (int a[],int size,int search){
  int beg=0,end=SIZE-1,mid;
  While ( beg<=end){
    mid =(beg+end)/2;
    if (a[mid]==search){
         return mid;}
    else if (a[mid]<search){
         return end=mid-1;}
    else{beg = mid+1;
    }
   }return -1;}

int main(){
  int a[100], size, search,i,pos=-1,loc1,loc2;
  printf("Enter the size of the array(max 100)");
  Scanf("%d", &size);
 printf("Enter elements in array:");
 for(i=0;i<size;i++)
    {scanf("%d",&a[i]);
 }
qsort(a,size,sizeof(int),comparator);
printf("sorted array is:");
for(i=0;i<size;i++)
       printf("%d",a[i]);
printf ("Enter search element");
 scanf ("%d", & search);
pos = binary search (a, size, search);
 if (pos == -1).
     printf ("Not found");
else
    printf (" The %d search element index : %d \n", search, pos);
printf(" Enter two indexes : ");
scanf ( " %d %d , & loc1, & loc2);
printf (" Sum is :", a[i+1] + a[i]);
 printf ("Product is %d", a[loc1], a[loc2]);
 return 0;
}.
```

Output

```
Enter the size : 6
 Enter elements in array :-
  1. 5  4  3   2  6

 Sorted array :-
 1   2  3  4  5  6

Enter search element : 3

 The search element found at index 2
Enter two index : 2 4
Sum is 8

Product = 15
```

```
2 #include <stdio.h>
#define SIZE 100
int a[SIZE];
void merge [int i, i1, int i2, int v2){
int i, j, k, temp [SIZE];
  v=0;
  i=i1;
  j=i2;
while((i<=i1)&&(j<=i2))
{
    if (a[i]<a[j])
      { temp[K]=a[i], i++, k++;
      }

  else {temp[v]=a[j]; j++; k++;
       }}
    while (i<=i1){
        temp [v] = a[i]; i++, k++;
    }
    while (j<= i2)&
     { temp[v]=a[j]; j++, k++;
  }
for (i=1; k=0; i<i2; i++, k++)
    { a[i]=temp[k];
       }}
void mergesort (int lb, int ub) {
    if(lb < ub){
        int mid =(lb+ub)/2
      mergesort (lb, mid);
      mergesort (mid+1, ub);
      merge( lb, mid, mid+1, ub); }}
```

```
int main (){
int i, n, product =1,k;
  printf(" Enter the size of array:");
  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");
  scanf("%d",&k);
for (i=0;i<k;i++)
  {  product = a[i];
  }
printf(" The product of the element is %d", product);
return0;
 }
```

Output

```
Enter size: 4
a[0] = 1
a[1] = 0
a[2] = 3
a[3] = 20
Enter k : 2
The product = 3.
```

Suppose an array A with n elements A[1], A[2]......A[N] is in memory. The insertion sort algorithm scans A from A[1] to A[N]; insertion each element [A[k] into its proper position in previous sorted subarray A[1], A[2], A[3]...... A[k-1];

Example:—

Array initial: 39, 22, 55, 68, 21

~~pass 1 : 39, 22, 55, 68, 21~~

pass 1:- 22, 39, 55, 68, 21

pass 2:- 21, 39, 55, 68, 22

pass 3:- 21, 22, 55, 68, 39

pass 4:- 21, 22, 39, 68, 55

pass 5:- 21, 22, 39, 55, 68 Sorted.

Pseudo code:—

```
A[10] = min integer value
Repeat steps 2 through 8 for k=1, 2, 3, .....N-1
temp = A[k]
ptr = k-1
Repeat steps 6 to 7 while temp < A[ptr]
    {   A[ptr+1] = A[ptr]
        ptr = ptr-1
    }   A[ptr+1] = temp;
        }  END
```

Time complexity :-
Best o(n), Average $o(n2)$, worst $O(n2)$

Selection Sort

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

~~Pass 1:~~ Example :- 15, 6, 13, 3, 2 → smallest

Pass 1 :- 2, 15, 6, 13, 3 → smallest

Pass 2 :- 2, 3, 15, 6, 13 → smallest

Pass 3 :- 2, 3, 6, 15, 3 → smallest

Pass 4 :- 2, 3, 6, 13, 15

Pseudocode :

```
Small = AR[L]
 For I = L to U do{
   Small = AR[i], pos=I
   For J = I+1 to U do{
    If AR[j] < small then {
            small = AR[i], pos=[i];
                  }
         j = J+1;
 temp = AR[i], AR[i]= small, AR[pos]=temp)
      }END
```

Time complexity best : o(n)

Average : $O(n^2)$

Worst : $O(n^2)$

```
4  #include <stdio.h>
void displayAltSum (int a[ ], int size )
{  int i, sum=0 ; product =1;
 printf ("Alternate elements:");
  for( i=0; i< size ;i++)
   {  if (i%2 ==0) {
          product+ = a[i];
   }
  else { sum + =a[i];
  printf("%d ",a[i]);
  }}
   printf (" Sum of odd elements = %d", sum);
   printf ("Sum of even elements= %d, product);
void div (int a[ ], int size ){
 int i=0; m ;
  printf(" Enter m ");.
   scanf("%d", &m);
 printf ("Elements divisible by %d ", m);
 for (i=0 ; i = size -1 ; i++)
   if (a[ i]% m ==0)
          printf("%d", a[i]);
      }}
Void bubblesort (int a[ ], int size)
{ int i, j, temp;
 for (i=0; i<=size-1; i++)
   for(j=0; j< size -i -1; j++)
     if (a [j]>a[j+1]) {
          temp = a[j];
          a[j] =a[j+1];
          a[j+1]=temp;    }
~~Print~~ display Altsum Pro (a,size);
  div (a,size );
 } int main ( )
 {  int a [100], size,i;
 printf (" Enter size of array :");
 scanf (" %d ", & size);
  printf(" Enter elements in array: ");
 for(i=0 ; i<size ; i++)
 {  scanf ( " %d" ,&a [i] );
  }
   bubble sort (a, size-1);
  Return 0;
 }
```

5

```
#include <stdio.h>
int binary search (int a[ ], int beg, int end, int search ){
int mid;
if (beg <= end ){
mid = (beg+ end)/2;
if ( a[mid] == search ) return mid;
if(a[mid > search)
        return binary search (a, beg, mid-1, search);
Return binary search (a, mid+1, end, search);
 }
 return -1;
}
int main()
{
  int a[100], size , search , i, pos;
  printf ( "Enter the size of array: ");
  scanf ("%d" ,&size );
  printf ("Sorted elements :");
for(i =0; i<= size; i++)
  {
    scanf ("%d" ,&a[i]);
  }
 printf ("Enter search element ");
scanf ("%d" ,&search);
pos = binarysearch (a, 0, size -1, search);
 if (pos == -1) { printf ("Not found ");
else{ printf ("Search element found at index %d ", pos);
  return 0;}
  }
```

Output :-

```
Enter size : 5
Enter sorted elements :- 1 2 3 4 5
Enter search element =2
Search element found at index 1.
```