Kondamuri Lahari
AP 19110010548
CSE-G

Assignment-6

1. Take the elements from the user and sort them in descending order and do.

a) Binary search find the element and location in the array.

b) Ask the user to enter any two locations Print the sum and Product of values at those locations in the sorted array.

Input:

```
#include<stdio.h>
#define NUM 30.
void bubblesort (int array[ ], int size)
{
for (int i=0; i<size-1; i++)
for (int j=0;j<size-i-1;j++)
{
if (array[j]<array[j+1])
{
int temp=array[j]
array[j] = array[j+1]
array[j+1] = temp;
}
}
}
void display (int array[], int size)
{
```

```
for (int i=0; i<size; i++) {
 printf ("%d", array[i]);
}
printf ("\n");
}
int binarysearch (int array[], int l, int r, int x) {
 if (r>=1){
    int mid = l+(r-1)/2;
    if array [mid] == x){
      return mid;
    }.
    else if (array[mid]>x){
      return binary search (array, l, mid-1, x);
    }
    else{
      return binary search (arrays, mid+1, r, x);
    }
    }
    return -1;
  }
void suman and product (int array[]){
   int loc1, loc2;
   printf("Enter location 1: ");
   scanf ("%d", &loc1);
   printf ("Enter location 2: ");
   scanf ("%d", &loc2);
   printf ("sum of elements in positions %d and %d is
        %d \n", loc1, loc2, array [loc1-1] + array [loc2-1]);
```

```
printf("product of elements in position %d and %d is:
    %d \n", loc1, loc2, array[loc1-1]*array[loc2-1];
}
int main()
{
  int a[NUM], size, k, r, result;
  printf("Enter no of elements of array:");
  scanf("%d", &size);
  for(k=0; k<size; k++)
  {
   printf("Enter the %dth elements:", k+1);
   scanf("%d", &a[k]);
  }
  printf("Given array:\n");
  display(a, size);
   bubble sort(a, size);
   printf("Sorted array in descending order:\n");
   display(a, size);
   printf("a:)\n");
   printf("Enter the element to search:");
   scanf("%d", &r);
   result = binary search(a, 0, size-1, r);
   if(result==-1){
     printf("%d element is not found in sorted array
                                        \n", r);
  }
  else{
     printf("%d element is found in sorted array at
          location %d \n", r, result+1);
  }
```

printf("b)\n");
sum and product(a);
return 0;
}

output:

Enter no of elements of array : 3
Enter the 1st element : 12
Enter the 2nd element : 56
Enter the 3rd element : 78
Given array
12 56 78
Sorted array in descending order:
78 56 12
a) Enter the element to search: 12
12 element is not found in sorted array.
b)
Enter location 1: 12
Enter location 2: 56
Sum of elements in Position 12 and 56 is : 32688
Product of elements in positions 12 and 56 is : 0

2. Sort the array using Merge sort where elements are taken from the user and find the product of kth element from first and last where K is taken from the user.

```
#include <stdio.h>
# define MAX 100
```

~~void merge arrays (int arr1[ ], int arr2[ ], int arr3[ ], int n1, int n2~~

```
int a[ms];
void merge (int l1, int l2, int u1, 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++;
      }
    }
    while (i<=u1) {
      temp[k] = a[i]; i++; k++;
    while (j<=u2) {
      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 Kth element is %d \n",
                                        product);
   return 0;
}
```

output:

Enter the the size of the array max (100):

a[0] = 1

a[1] = 5

a[2] = 12

a[3] = 21

Enter K

15

The product till the K th element is 21

3. Insertion sort:

Defination: Insertion sort works by inserting the set of values in the existing sorted file. It constructed the sorted array by inserting a single element at a time. this process continuous until whole array is sorted in same order. The primary concept behind insertionsort is to insert each item into its appropriate place in the final list, The insertion sort method save an effective amount of memory.

Advantages:

→ Easily implemented and very efficient when used with small sets of data.

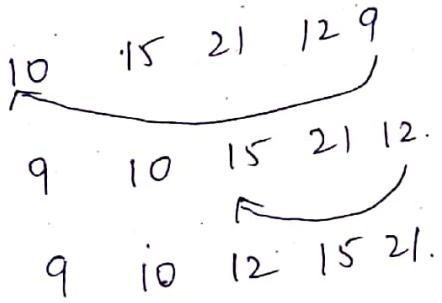
→ The additional memory space recuirment of insertion sort is less. i.e (O(1)).

→ It is considered to be live sorting technique as the list can be sorted as the new elements are received.

→ It is fasten than other sorting techniques.

Example:

array initial : 10 15 21 12 9.



Time complexity.

best : O(n)

average O(n²)

worst O(n²).

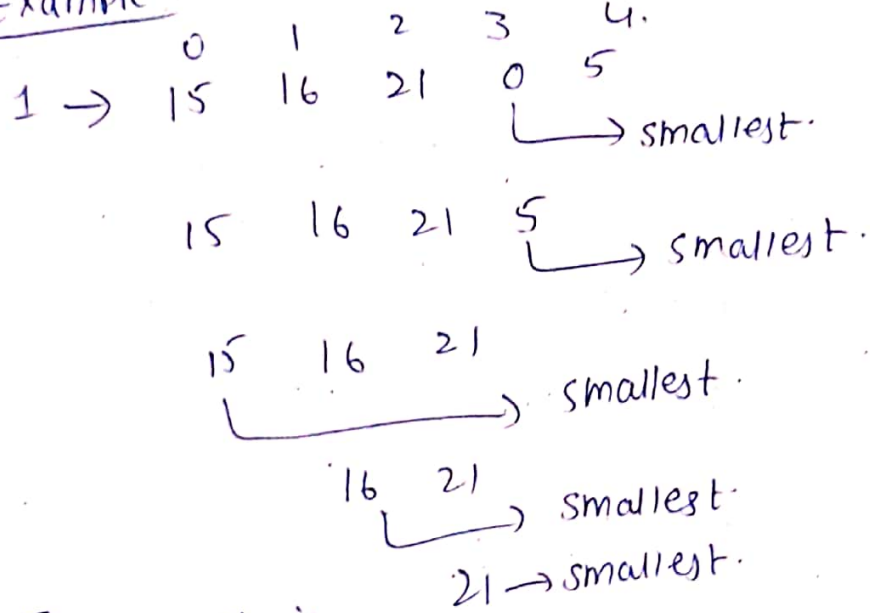
Selection sort: The selection sort perform sorting by searching for the minimum value number and placing it into the first of last position according to the order. The process of searching the minimum key and placing it in the proper position is continued untill the all elements are passed at right position.

Advantages

→ suppose an array ARR with N elements in the memory

→ simple to understand the sorting of elements doesn't depend on the initial arrangement of the Elements.

Example:



Time complexity:

best (O(n))

worst $O(n^2)$

average $O(n^2)$.

4. sort the array using bubble sort.

i) alternate order.

ii) sum of elements in odd positions and product.

iii) Elements which are divisible by m is taken.

Input:

```
#define <stdio.h>
#define NUM 30
void bubblesort (int array [ ], intsize)
{
 for (int i=0; i<size-1; i++)
 for (int j=0;; j<size-i-1; j++)
{
  if (array[j]>array[j+1];
  {
   int temp = array[j];
    array[j] = array[j+1];
     array[j+1] = temp;
   }
  }
}
void display (int array [ ], int size)
{
 for (int i=0 ; i<size; i++) {
   Print ("%d", array[i]);
  }
  printf ("\n");
}
void alternat (int array[ ], int size)
{
 for (int i=0; i<size ; i = i+2) {
  printf ("%d", array[i]);
 } printf ("\n");
```

```
}
void sumandproduct (int array[], int size)
{
  int sum=0, product=1;
  for (int i=0; i<size; i=i+2){
    sum=sum+array[i];
  }
  for (int j=1; j<size; j=j+2){
    product = product*array[j];
  }
  printf ("Sum of elements in odd position: %d \n", sum);
  printf ("Product of elements in even position: %d \n", product);
}
void divisible (int array[], int size)
{
  int m;
  printf ("Enter the value of m: ");
  scanf ("%d", &m);
  printf ("Elements of array divisible by %d are: \n", m);
  for (int i=0; i<size; i++){
    if (array[i]%m==0){
      printf ("%d ", array[i]);
    }
  }
}
```

```
int main()
{
 int a[Num],size,k;
 Printf("Enter no of elements of array:");
 scanf("%d",&size);
 for(k=0;k<size;k++)
{
  printf("Enter the %dth element :",k+1));
   scanf("%d",&a[k]);
}
printf("Given array:\n");
display(a,size);
bubblesort(a,size);
 printf("Sorted Array in Ascending orden:\n");
display(a,size);
 Printf("Sorted Array in Alternet orden:\n");
 alternate(a,size);
  Printf("b)\n");
  sum and Produt(a,size);
   Printf("c)\n");
   divisible(a,size);
   return 0;
}
```

output:

Enter the elements of array: 4.
Enter the 1st element: 12
Enter the 2nd element: 24
Enter the 3rd element: 35
Enter the 4th element: 78

Given array
12 24 35 78.
Sorted array in Ascending order:
12 24 35 78.

a) Sorted array in Alternet order:
12 35.

b) Sum of elements in odd position: 47.
Product of elements in even position: 1872

c) Enter the value of m: 2.
Elements of array divisible by 2 are;
12 24 78.

5. Write a recursive program to implement binary search?

Input:

```
#include <stdio.h>
Void binary_search (int [ ], int, int, int);
Void bubble_sort(int [ ], int);
int main ( )
{
  int key,size,i;
  int list [25];
  Printf ("Enter size of a list :");
  scanf ("%d", &size);
  Printf ("Enter elements \n");
  for( i=0; i<size; i++)
  {
    scanf("%d", &list [i]);
  }
  bubble_sort (list,size);
  Printf ("\n");
  printf (" Enter key to search \n");
  scanf ("%d", &key);
  binary_search (list, 0,size, key);
}
void bubble_sort (int list [ ], int size)
{
  int temp, i, j;
  for (i=0; i<size ; i++)
  {
    for (j=i; j<size; j++)
```

```
{ if list[i] > list[j]
  {
    temp = list[i]
    list[i] = list[j]
    list[j] = temp;
  }
}
}
void binary_search(int list[], int lo, int hi, int key)
{
  int mid;
  if (lo > hi)
  { printf("key not found\n");
    return;
  }
  mid = (lo + hi) / 2
  if (list[mid] == key)
  { print("key found \n");
  }
```

```
}
else if (list[mid] > key)
{
  binary_search (list, lo, mid-1, key);
}
else if (list[mid] < key)
{
  binary_search (list, mid+1, hi, key)
}
}
```

output:

```
size of list: 4
elements:
2 4 6 8
Enter key to search.
4
key found.
```