D.S.A Assignment

K.Vivek
AP19110010413
CSE-F

1) Take the elements from the user and sort them in descending order & write the following

a) using Binary search find the elements and the location in the array where the element is asked from the user

b) Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array

Ans:
```
#include<stdio.h>
int main()
{
int i, low, high, mid, n, key, a[a], [100], tmp, j, one
two, sum product;
Printf ("enter the number of elements in array");
scanf ("%d", &n);
Printf("enter %d integers", n);
for (i = 0 ; i<n; i++)
scanf ("%d", &a[i]);
for (i=0; i<n; i++);
```

```
for (j=j+1; j<n; j++)
{
  if (arr[i] < arr[j])
  {
    temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }
}
}
printf ("\n elements of array is sorted in
    desecending order:\n")
for(i=0; i<n ;i++)
{
printf ("enter value to find");
scanf ("%d", &key);
low = 0;
high = n-1;
mid = (low + high)/2;
while (low <= high) {
if (arr[mid] > key)
low = mid+1;
else if (arr[mid] == key)
```

```
printf ("%d found at location %d\n", key, mid+1);
break;
}
else
high = mid-1;
mid = (low + high)/2;
}
if (low > high)
{
printf("Not found! %d isn't present in the list\n", key);
}
printf("\n");
printf("Enter two location of find sum and product of the element\n");
scanf("%d", &one);
scanf("%d", &two);
sum = (arr[one] + arr[two]);
Product = (arr[one] * arr[two]);
Printf("The sum of element is = %d", sum);
Printf("The Product of element is = %d", Product);
return 0;
}
```

output :

enter number of elements : array 5

enter 5 integers

7

5

2

element of array in sorted in descending order

9 7 5 4 2 enter value to find 5

5 found, at location 3

enter two location & to find sum and product of the elements 2

4

The sum of elements = 7

the product of elements = 10

2) sort the array using merge sort where elements are taken from the user and find the product of kth elements from first and sum where k+i is taken from the user

Any :=#include<stdio.h>

=#include<conio.h>

=#define max-size 5

void merge_sort(int,int)

void merge_array(int,int,int,int);

int arr[sort [max-size];

int main()

```
printf("simple merge sort Example functions on
        array \n");
printf("\n Enter %d elements for sig\n",
        MAX_SIZE);
for (i=0; i<MAX_SIZE; i++)
scanf("%d", &arr_sort[i]);
printf("\n our data:");
for (i=0; i<max_size; i++){
printf("%d", arr_sort[i]);
}
merge_sort(0, max_size-1);
printf("\n\n sorted data:");
for(i=0; i<max_size(i++){
Print("%d", arr_sort[i]);
}
Print("find the product of kth element from
first and last where k\n");
scanf("%d", &k);
pro=arr_sort[a]*arr_sort[MAX_SIZE-k];
Print("Product = %d", pro);
  get ch();
}
void merge_sort(int i, int j){
```

```
if(i<j){
m=(i+j){
  m=(i+j)/2;
 merge_sort (i,m);
merge_sort (m+1,j);

//merging two arrays

merge_array (i,m,m+1,j);
}
}

void merge_array(int a, int b, int c, int d){
int t[50];
int k=a, j=c, k=0;
while(i<=b && j<=d){
if(arr_sort [i] < arr_sort[j])
t[k++] = arr_sort[i++];
 else
t[k++] = arr_sort[j++];
 }

//collect remaining elements.
 while(i<=b)
t[k++] = arr_sort [i++];
 while (j<=d)
for (i=a, k=0; i<=d; i++, k++)
arr_sort [i] = t[k];
 }
```

output:

simple merge sort example - functions and Array.

enter 5 elements for sorting

9
7
4
6
2

unsorted data : 9 7 6 4 2

sorted data : 2 4 6 7 9

find the product of kth elements from first and last where k

2

Product = 36

3) Discuss insertion sort and selection sort with example

Ans:

insertion sort works by inserting the sorted values in the existing sorted file. It constructs the sorted array by inserting a single element at a time. This process continues until whole array is sorted in same order the array concept behind insertion sort is to each item into its appropriate place in the final list.

Working of insertion sort:

* It uses two sets of array where one stores the sorted data and other unsorted data

* lets assume there are 'n' number elements in the array initially the element with index 0 (LB = 0) exist in the sorted region elements are in the unsorted portion of the list

* the first element of the unsorted region has array index 1 (if LB = 0)

* After each iteration it chooses the first element of unsorted partition and inserts it into the proper place in the sorted set

Advantages of insertion sort:

* easily implemented and very efficient when used with small sets of data

* The additional memory space requirement 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 element are received

* It is faster than other sorting algorithm

Definition of selection sort:

The selection sort perform sorting by searching for the minimum value number & and placing it into the first or last position according to the order the process of searching the minimum key and placing it in the proper position is continued until the all elements are placed at right position

Working of the selection sort:

Suppose an array ARR with N elements in the memory

- In the second pass again the position of the smallest value is determined in the subarray

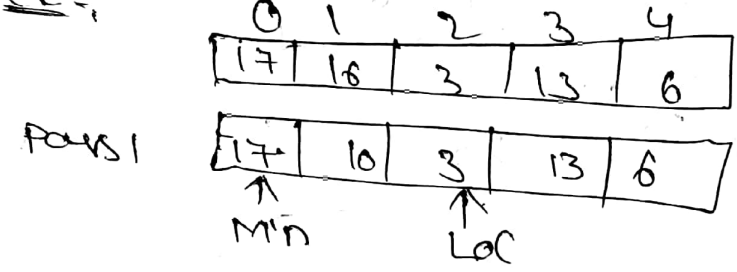
N-1 elements interchange the
ARR [1]

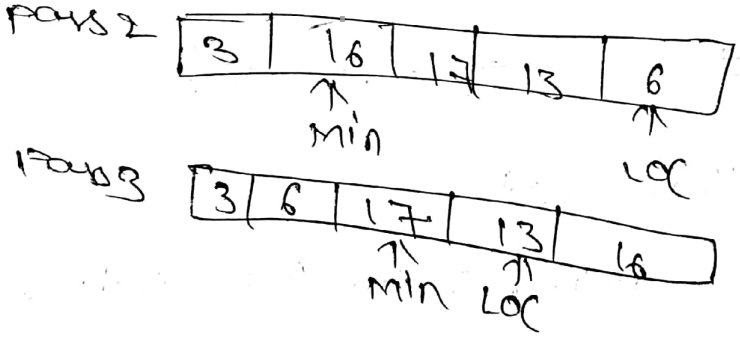
* In the pass N-1 the same process is perform
to sort all number of elements

Advantages of selection sort :-

* the main advantage of selection sort
is that it performs well on a small list

Ex :-





complexity of insertion sort :

the best case complexity of insertion sort is O(n) times. i.e when the array is previously sorted. In the same way when the array is sorted in reverse order the first element the unsorted set, so in the worst case running time of insertion sort is quadratic $O(n^2)$ in average case also it tray to make the minimum

$(k-1)/2$ comparisan, hence average quadratic has also running time (On^2)

complexity of selection sort :

As the working of selection sort does not depend on the original order of the element in the array, so there is not much difference between best case and worst case of complexity of selection sort

smallest element we require scanning of rest $n-1$ elements and the process is continued with the whole array sorted

$= (n-1) + (n-2) + \cdots + 2 + 1 = n(n-1)/2 = On^2$

4 sort array using bubble sort where elements are taken from the user and display the elements.

i) in ascending order

ii) sum of elements in odd positions and product of elements in even position

iii) elements which are divisible by m where m is take from the user

Ans:

```
#include <stdio.h>
#include <conio.h>
int main()
{
int arr[50], j, i, n, temp, sum = 0, product = 1;
printf("Enter total number of elements to stored");
scanf("%d", &n);
printf("Enter %d elements:", n);
for (i=0; i<n; i++)
scanf("%d", &arr[i]);
printf("\n sorting array using bubble sort \t\n");
for (i=0; i<(n-1); i++)
{
for (j=0 j<(n-1-1); j++)
{
if (arr[j] > arr[j+1])
{
temp = arr[j];
arr[j] = arr[j+1];
```

```
    }
  }
}
printf("\nAll array elements sorted successfully\n");
printf("\nArray elements in ascending order\n\n");
for(i=0;i<n;i++){
printf("%d\n",arr[i]);
}
printf("\narray elements in alternate order\n");
  for(i=0;i<n;i=i+2){
   printf("%d\n",arr[i]);
  }
for(i=1;i<=n;i+2){
  sum = sum + arr[i];
}
printf("\nThe sum of odd position elements
  = %d",sum);
 for(i=0;i<=n;i+2)
 {
 product = arr[i];
 }
printf("\nThe  Product of even positionelements
                are %d\n product);

    getch();
    return();
  }
```

output:

```
enter total number of elements to store;
5
Enter 5 elements; 8
 6
  4
   3
   2
```

sorting array using bubble sort technique

All array elements sorted successfully;

Array elements in ascending order;

```
  2
  3
  4
  6
  8
```

array elements in alternate order

```
 2
  4
  8
```

The sum of odd position element are=9

the product of even position elements are=64

5) write a recursive program to implement binary search;

```
#include <stdio.h>
#include <stdlib.h>
void Binary search(int arr [], int num, int first, int
last){
int mid;
if (first > last){
print ("number is not found");
}
else{
/* calculate mid element*/
mid = (first + last)/2;
// if mid is equal to number we are searching
if (arr [mid] == num){
Printf ("element is found at index %d", mid);
exit(0);
}
else if (arr [mid] > num){
Binary search (arr num, first, imid - 1);
} else{
Binary search (arr num, midt1, last);
}
```

```
void main(){
int arr[100],beg,mid,end,i,n,num;
printf("enter the size of an array");
scanf("%d",&n);
printf("enter the values in sorted sequence\n");
for(i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
beg=0;
end=n-1;
printf("enter a value to be search:");
scanf("%d",&num);
Binary search(arr,num,beg,end);
}
```

output:

```
enter the size of an array 5
enter the value in sorted sequence
4
5
6
7
8
enter a value to be search: 5
element is found at index 1
```