DSA ASSIGNMENT

M. Niteesh
AP19110010322
CSE-F

1) Take the elements from the user and sort them in decendingorder and do the following-

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

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

```
#include<stdio.h>
int binary search (int arr [], int x, int y,
                   int z)
{
if (y >= x)
{int mid = x + (y-x)/2;
if (arr [mid] == z)
return binary search (arr, x, mid-1, z);
return binary search (arr, mid+1, y, z);
}
return -1;
}
```

```
int main()
{
int num;
Printf("Enter size:");
scanf("%d", &num);
int l, m, x, val[num], OP, var, op1, op2, sum, pro;
for(x=0; x<num; x++)
 {
 Printf("Give value pls:");
 scanf("%d", &val[x]);
 }
for(l=0; l<num; ++l)
 {
for(m=0; m<num; ++m)
 {
 if(val[l]<val[m])
 {
   x = val[l];
   val[l] = val[m];
   val[m] = x;
 }
 }
Printf("That is in desending order:");
for(l=0; l<num; l++)
 {
 Printf("%d", val[l]);
 }
```

```
Printf ("\toptions\t\n");
Printf ("1. Find value at entered position\n");
Printf ("2. Find value at entered element\n");
Printf ("3. Print sum & Product of values at
        entered locations\n");
Printf ("Choose an option\n");
Scanf ("%d", &op);
Switch (op)
{
Case 1:
Printf ("Enter position value (index)");
Scanf ("%d", &var);
Printf ("The value at position %d is
        %d", var, val [var]);
break;
Case 2:
Printf ("Enter element to find position\n");
Scanf ("%d", &var);
int result = binarysearch (val, D, num, var);
(result == -1) ? Printf ("Element is been
                not find");
: printf ("Elemet Found at index %d",
          result);
return 0;
```

```
case 3:
printf("Enter any two index values plz:");
scanf("%d%d", &p1, &p2);
sum = val[p1] + val[p2];
printf("Sum = %d \n", sum);
printf("Product = %d", pro);
break;
}
```

---

2) sort the array using merge sort where elements are taken from user. + & k also.

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

void merge(int arr[], int a, int b, int c)
{
int l, m, k;
int c1 = b-a+1;
int c2 = c-b;
int A[c1], B[c2];
for(l=0; l<c1; l++)
{
A[l] = arr[a+l];
for(m=0; m<c2; m++)
B[m] = arr[b+1+m];
l=0;
m=0;
k=1;
while(l<c1 && m<c2)
```

```
{
if (A[i] < = B[m])
arr[k] = B[m];
    m++;
k++;
}
```

```
3
Void mergesort (int arr[], int a, int c)
{
if(a<c)
 {
 int b = a+(c-a)/2;
 mergo sort (arr, a, b);
 mergo (arr(a, b, c);
 }
}
```

```
void printarray(int A[], int size)
{
int i;
for(i=0; i<size; i++)
printf("%d", A[i]);
printf("\n");
}
int main()
{
int size, n;
printf("Enter array size: ");
scanf("%d", &size);
int val[size];
for(n=0; n<size; n++)
{
printf("Give value: ");
scanf("%d", &val[n]);
}
printarray(val, size);
mergesort(val, 0, size-1);
int m, f, a, p1, p2, temp;
printf("Give n value: ");
scanf("%d", &k);
p1=p2=1;
for(f=0; f<=k; f++)
{
temp=val[f];
p1 = temp * p1;
}
for(a=size-1; a>=k; a--)
{
temp=val[a];
p2 = temp * p2;
}
printf("%d %d", p1, p2);
}
```

3) Insertion sort:- It is a simple comparision based sorting algorithm. It inserts every array element into its proper position. In i-th iteration, previous (i-1) elements are already sorted, the i-th element (Arr[i]) is inserted into it's proper place in the previously sorted subarray. And this array can be easily computed.

Ex:- 12, 11, 13, 5, 6

let us loop for i=1 to 4

i=1, since 11 is small than 12, move 12 and insert 11 before 12.

11, 12, 13, 5, 6

i = 2. 11, 12, 13, 5, 6

i = 3. 5 will move to the beginning.

5, 11, 12, 13, 6

i = 4. 6 will move to position after 5.

5, 6, 11, 12, 13.

selectionsort:- It selects i-th smallest element and places at i-th position. This algorithm divides the array into two parts sorted (left) & unsorted (right) subarray. It selects smallest element from unsorted subarray and places in the first position of that subarray. It repeatedly selects the next smaller element.

Ex:-

~~4 2 3 1 5~~ ~~5 4 3 1 2~~ arr[] = 5 4 3 1 2

1. Find the min element in arr[0--4] and place it at begining

1, 5, 4, 3, 2

2. Find the min element in arr[1--4] and place it at begining of arr[1--4]

1, 2, 5, 4, 3

3. Find min element ~~in arr~~ and place it at begining of arr[2--4]

1, 2, 3, 5, 4

4. Find min element & Place it at begining of arr[3--4]

1, 2, 3, 4, 5.

---

4)

```
#include<stdio.h>
void Bubble sort (int arr[], int n)
{
int i, j, temp;
for (i=0; i<n-1; i++)
for (j=0; j<n-i-1; j++)
if (arr[j] > arr[j+1])
{
temp = arr[j];
arr[j] = arr[j+1]
arr[j+1] = temp;
}
}
int main ()
{
int size, j;
```

```
3. printf("Give size of array");
   scanf("%d", &size);
   for(i=0; i<size; i++)
   {
   printf("%d", &arr[i]);
   }
   printf("1. Show elements in alternate order\n");
   printf("2. Sum of odd positive elements &
                Product of even positive elements\n");
   printf("3. Show by m");
   int op, sum=0, pro=1, m;
   printf("Give choice");
   scanf("%d", &op);
   switch(op)
   {
   case 1:
   for(i=0; i<size; i+=2)
   {
   printf("%d", arr[i]);
   case 2:
   for(i=0; i<size; i+=2)
   {
   sum = sum + arr[i];
   }
   for(i=0; i< size; i+=2)
   {
   pro = pro * arr[i];
   }
   printf("%d %d", sum, pro);
   case 3:
   printf("Give m value");
   scanf("%d", &m);
   printf("numbers devisible by %d
                are : \n", m);
   for(i=0; i<size; i++)
```

```
{
if (arr[i]%2==0)
{
printf("%d", arr[i]);
}
}
```

5) write a recursive program to implement binary search?

```
#include<stdio.h>
int binarysearch (int x[], int a, int b, int c)
{
int mid = (a+b)/2;
if (a>b)
return -1;
if (x[mid]==c)
return mid;
if (a[mid]<c)
return binarysearch (x, a, mid+1, c);
else
return binarysearch(x, a, mid-1, c);
}
int main(void)
{
int x[100], size, pos, val;
Printf("Give array size");
scanf("%d", &size);
~~Printf~~ for(i=0; i<size; i++);
scanf("%d", &a[i]);
Printf("Give element to search");
scanf("%d", val);
pos = binary search(x, 0, size-1, val);
if (pos<0)
printf("%d", val);
else
printf("%d", val, pos+1);
}
```

THE END