DSA Assignment

N. Praneeth Babu
AP19110010383
CSE-H.

1)
```
#include<stdio.h>
int main()
{
 int i, low, high, mid, n, key, arr[100], tmp, j, one, two, s, p;
 printf("Enter the Number of Elements");
 scanf("%d", &n);
 printf("Enter %d integers", n);
 for(i=0;i<n;i++)
 {
    scanf("%d", &arr[i]);
  }
 for(i=0;i<n;i++)
 {
     for(j=i+1; j<n; j++)
      {
        if(arr[i]< arr[j])
         {
            tmp= arr[i];
            arr[i]= arr[j]
            arr[j] = tmp;
          }
       }
   }
 printf(" Elements in descending order ");
 for(i=0;i<n;i++)
 {
    printf("%d", arr[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", key, mid+1);
     break;
   }
   else
     high=mid-1;
     mid=(low+ high)/2;
   }
   if(low>high)
     printf(" Not found");
   printf("\n");
  printf(" Enter two locations to find sum and product of t
                                                 elemen
  Scanf("%d", &one);
  Scanf("%d", &two);
  Sum=(arr[one] + arr[two]);
  P=(arr[one] * arr[two]);
  printf(" The Sum of elements= %d", S);
  printf(" The product of elements = %d", P);
} return 0;
```

```
2) #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()
   {
       int i, k, pro=1;
       printf(" Simple Merge Sort Example functions and Arrays");
       printf("\n Enter %d Elements for Sorting", MAX_SIZE);
       for(i=0; i<MAX_SIZE; i++)
       {
          scanf("%d", &arr_sort[i]);
       }
       printf("\n ~~Your~~ Data:");
       for (i=0; i<MAX_SIZE; i++)
       {
           printf("\t %d", arr_sort[i]);
       }
      merge_sort(0, MAX_SIZE-1);
      printf("\n\n Sorted Data:");
     for (i=0; i<MAX_SIZE; i++)
     {
       printf("\t %d", arr_sort[i]);
   }
  printf(" find the product of kth elements from first and
          last where k \n");
  scanf(" %d", &k);
  pro = arr_sort[k] * arr_sort[MAX_SIZE-k-1];
```

```
 printf("Product = %d", Pro);
 getch();
}
Void merge-Sort (int i, int j)
 {
   int m;
   if(i<j)
    {
      m=(i+j)/2;
      merge-Sort (l, m);
      merge-Sort (m+1, j);
      merge-array (i,m,m+1, j);
    }
 }
Void merge-array (int a, int b, int c, int d)
 {
   int t[50];
   int i=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++];
    }
   while (i<=b)
      t [k++]= arr-Sort[i++];
   while (j<=d)
      t[k++]= arr-Sort[j++];
   for (i=0; j=0; i<=d; i++; j++)
   { arr-Sort [i]= t[j];
 }
```

Scanned with CamScanner

3) The Selection Sort algorithm Sorts an array by repeatedly finding the minimum element (Considering ascending order) from unsorted part and putting it at the beginning.

The.

1) The Subarray which is already sorted.

2) Remaining Subarray which is unsorted.

In every Iteration of Selection Sort, the minimum element from the unsorted Subarray is picked and moved to the Sorted array.

[11:26 am, 04/05/2020]

arr[]= 64 25 12 22 11

// Find the minimum element in arr[0....4]
// and place it at beginning.
// 25 12 22
11 ~~25~~ ~~12~~ ~~22~~ 64

// Find the minimum element in arr[1....4].
// and place it at beginning of arr[1...4]

// 12 25 22 64

// Find the minimum element in arr[2... 4]
// and place it at beginning of arr[2...4].

~~1~~ 2

// 12 22 25 64

[11:26 am, 04/05/2020].

Insertion Sort is a simple Sorting algorithm that works the way we Sort playing cards in our hands.

Algorithm.

// Sort an arr[] of Size n.

InsertionSort (arr, n)

Loop from i=1 to n-1:

.... a) Pick element arr[i] and insert it into sorted sequence arr[0....i-1].

[11: 26 am, 04/06/2020]

12, 11, 13, 5, 6

let us loop for i=1 (Second element of the array) to 4

i=1. Since 11 is Smaller than 12, move 12 and insert 11 before 12

11, 12, 13, 5, 6.

i=2, 13 will remain at its position as all elements in A[0...i-1] are Smaller than 13.

11, 12, 13, 5, 6.

i=3. 5 will move to the beginning and all other elements from 11 to 13 will move on position ahead of their current position.

5, 11, 12, 13, 6.

i= 4. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.

5, 6, 11, 12, 13.

```
1) #include<stdio.h>
void main()
{ int a[100], n, i, j, temp, sumo=0, proud=1, m;
  printf(" Enter number of elements \n");
  scanf(" %d", &n);
  printf(" Enter %d integers\n", n);
  for(i=0; i<n; i++)
  {
    ~~for~~ scanf(" %d", &a[i]);
  }
  for(i=0; i<n-1; i++)
  {
    for(j=0; j<n-i-1; j++)
    {
      if(a[j]>a[j+1])
      {
        temp=a[i];
         a[i] = a[j+1];
         a[j+1] = temp;
      }
    }
  }
  printf(" \n Sorted list in ascending order: \n");
  for(i=0; i<n; i++)
  {
    printf(" %d \n", a[i]);
  }
  printf(" The alternate order is ");
```

```
for(i=0; i<n; i++)
 {
   if(i% 2==0)
   {
      printf ("%d", a[i]);
   }
 }
for(i=0;i<n;i++)
{
   if(i%2!=0)
   {
      Sumo=Sumo+a[i];
   }
}
printf("\nSum of odd index is %d", Sumo);
for(i=0; i<n; i++)
{
  if (i%2==0)
  {
     prod = prod * a[i];
  }
}
printf("\n Product of odd index is %d", prod);
printf("\n Enter the value of m \n");
Scanf ("%d", &m);
for(i=0; i<n; i++)
{
  if(a[i]%m==0)
  {
     printf ("%d ", a[i]);
  }
}
```

9)

```
#include <Stdio.h>
int recursiveBinarySearch (int array[], int Start-index, int end-index,
                                   int element)
{
  if(end-index >= Start-index)
  {
   int middle = Start-index + (end-index - Start-index)/2;
   if (array[middle] == element)
      return middle;
   if (array[middle] > element)
      return recursiveBinarySearch (array, Start-index, middle-1,
                                                  element);
   return recursiveBinarySearch (array, middle+1, end-index, element);
  }
  return -1;
}
int main (void)
{
  int array[] = {1, 4, 7, 9, 16, 56, 70};
  int n=7;
  int element =9;
  int found-index = recursiveBinarySearch (array, 0, n-1, element);
  if (found-index == -1)
  {
    printf("Element not found in the array");
  }
  else
  {
   printf (" Element found at index : %d", found-item);
  }
  return 0;
}
```