Assignment-6

K. Sai Abhinav
CSE-F
AP19110010339

1)
```
#include<stdio.h>
void binary(int [], int, int, int); /* declaring two functions starting function is to sort the elements in the array */
void sorting (int[], int); /* binary function is to implement binary search in array */
int main()
{
  int num, length, i, x, y, sum, product;
  /* Initialising the array and declaring variables */
  int arr[100];
  printf("Enter the length of array : ");
  scanf("%d", &length);
  printf("Enter elements in array \n : ");
  for(i=0; i< length; i++)
  {  scanf("%d", &arr[i]);
  }
  sorting(arr, length); /* we are calling the soting function to sort the elements in array */
  printf("\n");
  printf("Enter number to search in array\n");
  scanf("%d", &num);
  binary(arr, 0, length, num);
```

```
/* by using binary function we are searching
the element given by user */
}
void sorting(int arr[], int length)/*Inside the
                                    function*/
{
  int temp, i, j, sum, product, x, y;
    for(i=0; i< length; i++)
    {
      for(j=1; j<length; j++)
      {
        if(arr[i] > arr[j]) /* By using this logic
        {                      are sorting the elem
          temp = arr[i];       in descending order
          arr[i] = arr[j];
          arr[j] = temp;
    } } }
  printf("Sorted array is : \n");
  for(i=0; i< length; i++)
  { printf("%d\t", arr[i]);
  }
  printf("Enter 1st position:\n");
  scanf("%d", &x);
  printf("Enter 2nd position:\n");
  scanf("%d", &y);
  if(x > length || y > length)
  {
    printf("Enter Valid positions");
  }
```

```
else
{
  sum = arr[x] + arr[y];        /* we are calculating sum
  printf("sum = %d", &sum);      and product of particular
  product = arr[x] * arr[y];     two positions in array
  printf("product = %d", product);  given by user */
}
}
void binary(int arr[], int x, int y, int num)
{ int mid;
  if (x > y)
  {
    printf("\nNumber is not found in array");
  }
  mid = (x+y)/2;
  if (arr[mid] == num)          /* By using this logic we
  {                              are implementing binary
    printf("Number is found in array\n");   search */
  }
  else if (arr[mid] > num)
  {
    binary(arr, x, mid-1, num);
  }
  else if (arr[mid] < num)
  {
    binary(arr, mid+1, y, num);
  }
}
```

2)

void

```
#include<stdio.h>
void merge_sort(int a[], int i, int j);
void merge(int a[], int i1, int j1, int i2, int j2);
/* By using merge sort we are dividing the array
into two halves we are sorting the two arrays
individually */
/* After completion of sorting of two arrays
we are merging the two arrays to get the
sorted array */
int main()
{
  int arr[100], n, i, k, product;
  printf("Enter number of elements in array: \n");
  scanf("%d", &n);
  printf("Enter elements in array: ");
  for(i=0; i<n; i++)
  {
    scanf("%d", &arr[i]);
  }
  merge_sort(arr, 0, n-1);
  printf("\n Sorted array is : ");
  for(i=0; i<n; i++)
  {
    printf("%d\t", arr[i]);
  }
  printf("Enter the value of k less than
    %d", n);
```

```
scanf("%d", &k);
product = arr[k] * arr[n-k];
printf("\n Product of two elements is %d",
                                product);
}
void merge_sort(int arr[], int i, int j)
{ int mid;
  if(i<j)
  { mid=(i+j)/2;
    merge_sort(arr, i, mid);
    merge_sort(arr, mid+1, j);
    merge(arr, i, mid, mid+1, j);
  }
}
void merge(int arr[], int i1, int j1, int i2, int j2)
{ int temp[100], i, j, k;
  i=i1, j=i2, k=0;
  while(i<=j1 && j<=j2)
  { if(arr[i]<arr[j])
    { temp[k++] = arr[i++];
    }
    else
    {
      temp[k++] = arr[j++];
    }
  while(i<=j1){
        temp[k++] = arr[i++]; }
  while(j<=j2){
      temp[k++] = arr[j++]; }
  for(i=i1, j=0; i<=j2; i++, j++){
        arr[i] = temp[j]; }
```

3) Insertion sort and selection sort:-

Insertion sort:-

Insertion sort is a sorting algorithm where the array is sorted by taking one element at a time. The pricipla behind the insertion sort is to take one element, iterate through the sorted array and find its correct position in the sorted array. Insertion array works in a similar way as we arrange the deck of cards.

Algorithm:-

step 1: If the element is the first one, it is already sorted.

step 2: Move to the next element.

step 3: compare the current element with all the elements in the sorted array.

step 4: If the element in the sorted array is smaller than the current element, iterate to the next element. otherwise, shift all the greater element in the array by one position towards the right.

Step 5: Insert the value at the correct position

Step 6: Repeat until the complete list is sorted.

As the average and worst case complexity of this algorithm are $O(n^2)$ where n is the no. of elements, insertion sort is not good for large data sets.

Selection Sort:-

Selection sort is the most conceptually simple of all the sorting algorithms. It works by selecting the smallest element of the array and placing it at the head of array. Then the process is repeated for the remainder of array; the next largest element is selected and put into the next slot, and so on down the line. Because a selection sort looks at progressively smaller parts of array each time, a selection sort is slightly faster than bubble sort and can be better than a modified bubble sort.

Examples:

For sorting the array 52314 first, 2 is inserted before 5, resulting in 25314 Then, 3 is ~~inster~~ inserted between 2 and 5; resulting in 23514, one is inserted at start, 12354 finally, 4 is inserted between 3 and 5, 12345

Time complexity:

O(n2) as there are two nested loops -

4) #include<stdio.h>

```
int main() {
 int arr[50], n, i, j, temp, sum=0, product=1, k;
  printf("Enter no. of elements in array\n");
  scanf("%d", &n);
  for(i=0; i<n; i++)
  {
     scanf("%d", arr[i]);
  }
  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;
      }
  } }
  printf("Sorted array in ascending order is\n");
  for(i=0; i<n; i++)
  {
     printf("%d\n", arr[i]);
  }
  printf("Sorted array in alternate order is\n");
  for(i=0; i<n; i=i+2)
  {
     printf("%d\n", arr[i]);
  }
```

```
printf("Sum of all elements in
are :\n");
for (i = 0; i<n; i = i+2)
{
    sum = sum + arr[i];
    printf ("%y.d\n", sum);
}
printf(" Product of all elements in even
position :\n");
for(i=0.1; i<n; i=i+2)
{
    product = product * arr[i];
    printf("%d\n", product);
}
printf("Enter a number :");
scanf("%d", &k);
printf("Elements divisible by %d are:\n", k);
for(i=0; i<n; i++)
{  if(arr[i] % k == 0)
   {
      printf("%d\n", arr[i]);
   }
}
return 0;
}
```

5) #include<stdio.h>
void binary(int [], int, int, int);
void sorting (int[], int);
int main() {
int num, length, i;
int arr[100];
printf("Enter length of array:");
scanf("%d", &length);
printf("Enter elements for array\n");
for (i=0; i< length ; i++)
{ scanf("%d", &arr[i]);
}
sorting (arr, length);
printf("Enter number to search:");
scanf("%d", &num);
binary (arr, 0, length, num);
}
void sorting(int arr[], int length)
{ int temp, i, j;
for (i=0; i< length; i++)
{ for(j=1; j< length; j++)
{
if(arr[i] > arr[j])
{
temp = arr[i];
arr[i] = arr[j];
} } } arr[j] = temp;

```
void binary (int arr[], int x, int y, int num)
{
  int mid;
  if(x>y)
  {
    printf("\nNumber is not found");
  }
  mid=(x+y)/2;
  if(arr[mid]==num)
  {
    printf("\nNumber is found");
  }
  else if(arr[mid]>num)
  {
    binary(arr, x, mid-1, num);
  }
  else if(arr[mid]<num)
  {
    binary(arr, mid+1, y, num);
  }
}
```