Mehedi Hasan Munna

191 –15–12946

1. Binary search

```
#include <stdio.h>
int binary search (int arr[], int l, int r, int x)
    { if (r >= l) {
            int mid = l + (r-1)/2;
            if (arr[mid] == x)
                return mid;
        if (arr[mid] > x)
                return binary search (arr, l, mid-1,
                                                    x);
        return binary search (arr, mid +1, r, x);
            } return -1;
}
int main (void)
{ int arr[] = {2,3,4,10,40};
    int n = sizeof (arr) / sizeof (arr[0]);
    int x =10;
    int result = binary search (arr, 0, n-1, x);
    (result == -1);
    printf (" Element is not present in array ");
    print f (" Element is present at index %d ",
                                            result);
    return 0;
}
```

2. Insertion Sort

```c
#include <math.h>
#include <stdio.h>
void intertion sort (int arr[], int n)
    { int i, key, j;
        for (i=1; i<n; i++) {
            key = arr[i];
            j = i-1;
        while (j >=0 && arr[j] > key) {
            arr[j+1] = key; ? arr[j];
            j = j-1;
        }
            arr[j+1] = key;
    }
}
void print Array (int arr[], int n)
    { int i;
        for (i=0; i<n; i++)
            printf ("%d", arr[i]);
        printf ("\n");
}
int main ()
{
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof (arr)/sizeof (arr[0]);
    insertion sort (arr, n);
    print Array (arr, n);
    return 0;
}
```

3. Selection sort

```c
#include <stdio.h>
void swap (int *xp, int *yp)
{ int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void selection sort (int arr[], int n)
{
    int i, j, min-idx;
    for (i=0; i<n-1; i++)
    { min-idx =i;
        for (j=i+1; j<n; j++)
        if (arr [j] < arr [min-idx])
            min_idx =j;
        Swap (&arr [min-idx], &arr [i]);
    }
}
void PrintArray (int arr[], int size)
{ int i;
    for (i=0; i<size; i++)
        printf ("%d", arr [i]);
    print ("\n");
int main ()
{ int arr[] = {64,25,12,22,11};
    int n= sizeof (arr)/ sizeof (arr[0]);
    selection sort (arr, n);
```

```c
printf ("sorted array : \n");
print Array (arr, n);
return 0;
}

4. Merge sort
#include <stdio.h>
#include <stdlib.h>
void merge (int arr[], int I, int m, int r)
{ int i, j, k;
  int n1 = m - I + 1;
  int n2 = r - m;
  int L[n1], R[n2];
  for (i = 0; i < n1; i++)
      L[i] = arr[I + i];
  for (j = 0; j < n2; j++)
      R[j] = arr[m + 1 + j];
  i = 0; // Initial index of first subarray
  j = 0; // Initial index of second subarray
  k = I; // Initial index of merged subarray
  while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
          arr[k] = L[i];
          i++;
      }
```

```
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    } while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void merge sort (int arr[], int 1, int r)
{ if (1 < r) {
        int m = 1 + (r - 1) / 2;
        merge sort (arr, 1, m);
        merge sort (arr, m+1, r);
        merge (arr, 1, m, r);
    }
}
void printArray (int A[], int size)
{ int i;
    for (i = 0; i < size; i++)
```

```c
        printf("%d", A[i]);
        printf("\n");
}
int main()
{   int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr)/sizeof(arr[0]);
    printf("Given array is \n");
    printArry(arr, arr_size);
    mergesort(arr, 0, arrsize - 1);
    printf("\n sorted arry is \n");
    printArray(arr, arr_size);
    return 0;

}
```

5. Quick sort
```c
#include <stdio.h>
void swap(int *a, int *b)
    { int t = *a;
    *a = *b;
    *b = t;
    }
int partition(int arr[], int low, int high)
    { int pivot = arr[high]; // pivot
        int i = (low - 1);
        for(int j = low; j <= high - 1; j++)
        { if(arr[j] < pivot)
            {
```

```c
            swap (&arr[i], &arr[j]);
        }
    }
    swap (&arr[i+1], &arr[high]);
    return (i+1);
}
void quicksort (int arr[], int low, int high)
{   if (low < high)
    {   int pi = partition (arr, low, hight);
        quicksort (arr, low, pi-1);
        quicksort (arr, pi+1, high);
    }
}
void printArray (int arr[], int size)
{   int i;
    for(i=0; i < size; i++)
        printf("%d", arr[i]);
    printf("n");
}
int main()
{
    int arr[] = {10,7,8,9,1,5};
    int n = sizeof(arr)/ sizeof(arr[0]);
    quicksort(arr,0, n-1);
    printf("sorted array : n");
    printArray (arr, n);
    return 0;
}
```