//sorting.h

    #include<stdlib.h>

    #define TRUE 1;

    #define FALSE 0;

    void bubbleSort(int\*, int);

    void insertionSort(int\*, int);

    void selectionSort(int\*, int);

    void mergeSort(int\*,int,int);

    void heapify(int\*,int,int);

    void heapSort(int\*,int);

    void swap(int\*, int\*);

    void merge(int\*,int,int,int);

    void useMergeSort(int\*,int);

    void bubbleSort(int\* arr, int n){

        for(int i = 0; i < n-1; i++){

            int swapped = FALSE;

            for(int j = 0; j < n-1-i; j++)

                if(arr[j] > arr[j+1]){

                    swap(arr+j,arr+j+1);

                    swapped = TRUE;

                }

            if(!swapped)

                break;

        }

    }

    void insertionSort(int\* arr,int n){

        for(int i = 1; i < n; i++){

            int key = arr[i];

            int j = i - 1;

            while(j >= 0 && arr[j] > key){

                arr[j+1] = arr[j];

                j--;

            }

            arr[j+1] = key;

        }

    }

    void selectionSort(int\* arr, int n){

        int minIndex ;

        for(int i = 0; i < n-1; i++){

            minIndex = i;

            for(int j = i+1; j < n; j++)

                if(arr[j] < arr[minIndex])

                    minIndex = j;

            swap(arr+minIndex,arr+i);

        }

    }

    void merge(int \*arr, int p, int q, int r){

        int n1 = q - p + 1;

        int n2 = r - q;

        int \*left,\*right;

        left = (int\*)malloc(n1\*sizeof(int));

        right = (int\*)malloc(n2\*sizeof(int));

        for(int i = 0; i < n1; i++)

            left[i] = arr[p+i];

        for(int i = 0; i < n2; i++)

            right[i] = arr[q+i+1];

        int i = 0, j = 0, k = p;

        while(i < n1 && j < n2){

            if(left[i] <= right[j]){

                arr[k] = left[i];

                i++;

            }

            else{

                arr[k] = right[j];

                j++;

            }

            k++;

        }

        while(i < n1){

            arr[k] = left[i];

            i++;

            k++;

        }

        while(j < n2){

            arr[k] = right[j];

            j++;

            k++;

        }

    }

    void mergeSort(int \*arr,int p, int r){

        if(p < r){

            int q = p + (r-p)/2;

            mergeSort(arr,p,q);

            mergeSort(arr,q+1,r);

            merge(arr,p,q,r);

        }

    }

    void useMergeSort(int \*arr,int n){

        mergeSort(arr,0,n-1);

    }

    void heapify(int\* arr,int n, int i){

        int max = i;

        int l = 2\*i + 1;

        int r = 2\*i + 2;

        if(l < n && arr[l] > arr[max])

            max = l;

        if(r < n && arr[r] > arr[max])

            max = r;

        if(max != i){

            swap(arr+i,arr+max);

            heapify(arr,n,max);

        }

    }

    void heapSort(int\* arr, int n){

        for(int i = n/2 - 1; i >= 0; i--)

            heapify(arr,n,i);

        for(int i = n-1; i > 0; i--){

            //delete the root and append to last position

            swap(arr,arr+i);

            heapify(arr,i,0);

        }

    }

    void swap(int\* a, int\* b){

        int temp = \*a;

        \*a = \*b;

        \*b = temp;

    }

**HELPER.H**

    #include<stdio.h>

    #include<time.h>

    #include<stdlib.h>

    void inputArray(int\*,int);

    void inputSize(int\*);

    void outputSortedArray(int\*,int);

    void outputTimeTaken(clock\_t,clock\_t);

    void useAlgo(void (\*algo)(int\*,int),int n);

    void useAlgo(void (\*algo)(int\*,int),int n){

        clock\_t start,end;

        int \*arr;

        arr = (int\*)malloc(n\*sizeof(int));

        inputArray(arr,n);

        start = clock();

        (\*algo)(arr,n);

        end = clock();

        outputSortedArray(arr,n);

        outputTimeTaken(start,end);

        free(arr);

    }

    void inputArray(int\* arr,int n){

        printf("\nEnter %d integers: ",n);

        for(int i = 0; i < n; i++)

            scanf("%d",arr+i);

    }

    void inputSize(int \*n){

        printf("\nArray Size : ");

        scanf("%d",n);

    }

    void outputSortedArray(int\* arr,int n){

        printf("\n");

        for(int i = 0; i < n; i++)

            printf("%d ",arr[i]);

    }

    void outputTimeTaken(clock\_t start,clock\_t end){

        float cpu\_time;

        cpu\_time = ((float)(end-start))/CLOCKS\_PER\_SEC;

        printf("\nTime taken in ms is %f \n",cpu\_time\*1000);

    }

**MAIN.C**

    #include<stdio.h>

    #include"./sorting.h"

    #include"./helper.h"

    #include<stdlib.h>

    int main(){

        int n;

        inputSize(&n);

        printf("BUBBLE SORT");

        // useAlgo(bubbleSort,n);

        printf("\nINSERTION SORT");

        useAlgo(insertionSort,n);

        printf("\nSELECTION SORT");

        useAlgo(selectionSort,n);

        printf("\nMERGE SORT");

        useAlgo(useMergeSort,n);

        printf("\nHEAP SORT");

        useAlgo(heapSort,n);

        return 0;

    }

**GRAPHS**