Submitted By - Sanyam Rajpal
CSCI-B-505 Applied Algorithms
(FA19 - BL - 11503)
Assignment - I
SOURCE CODE

```
// Since these are the common functions for all the different main functions for all plots, I wrote them
once on the top.
//including required libraries
#include <iostream>
#include <vector>
#include <algorithm>
#include <time.h>

// to not use std again and again when needed
using namespace std ;

// function to print array
void print_array(  vector< int > input  ){
    for( auto i : input )
        cout << i << " " ;
    cout << endl ;
}

// insertion sort algorithm implemented as a function
void insertion_sort( vector< int > input , int size ){
    int j , key ;
    for( int i = 1 ; i < size ; ++i ){
        key = input[ i ] ;
        j = i ;
        // ( j-- ) is true for all values of j greater than 0. Plus it narrows down the loop line by 1 by doing the
decrement
        while( ( j-- ) && ( input[ j ] > key ) )
            input[ j + 1 ] = input[ j ] ;
        input[ j + 1 ] = key ;
    }
    // cross checking that the array has been sorted
    cout << "Sorted Array" << endl ;
    print_array( input ) ;
}

// function to swap values of the array elements
// * is uses so that address is not changed and only the values at those addresses are changed which is
what we want
// the value of array elements are changed
void swap( int *x_pointer , int *y_pointer ){
    int temp = *x_pointer ;
    *x_pointer = *y_pointer ;
    *y_pointer = temp ;
```

```
}

// selection sort algorithm implemented as a function
void selection_sort( vector< int > input , int size ){
    int min_idx , j ;
    for( int i = 0 ; i < ( size - 1 ) ; ++i ){
        min_idx = i ;
        for( j = i + 1 ; j < size ; ++j ){
            if( input[ j ] < input[ min_idx ] )
                min_idx = j ;
        }
        swap( &input[ i ] , &input[ min_idx ] ) ;
    }
    // cross checking that the array has been sorted
    cout << "Sorted Array" << endl ;
    print_array( input ) ;
}

// Input-1 Plot-1 Main Function
// main function which calls the respective algorithm for sorting
int main()
{
    // initializing variables and arrays
    vector< int > input ;
    int size , j ;
    // asking user for input of the number of elements
    cout << "Give the number of elements you would like the randomly generated input to have." << endl ;
    cin >> size ;
    // initializing variables with values
    double sum_ins = 0.0 ;
    double sum_sel = 0.0 ;
    // time variables
    clock_t begin_ins , end_ins , begin_sel , end_sel ;
    for( int i = 0 ; i < 3 ; ++i ){
        // clearing input vector from the previous values stored so that a new set of random values can be added and later sorted to make 3 random sortings distinct
        // pushing size number of random values in range [ 1 , size ]
        input.clear( ) ;
        for( j = 0 ; j < size ; ++j )
            input.push_back( rand( ) % ( size ) + 1 ) ;
        // cross checking random arrays
        cout << "Random Array" << endl ;
        print_array( input ) ;
        begin_ins = clock( ) ;
        // here & operator isn't used so the sorted array won't be saved at the address we are accessing from input[ ] in the main function
        insertion_sort( input , size ) ;
        end_ins = clock( ) ;
        // calculating runtime of sorting and adding it to the sum of runtimes
        sum_ins += double( end_ins - begin_ins ) ;
        cout << "Execution Time for " << ( i + 1 ) << " time for insertion sort is " << double( end_ins - begin_ins ) <<
" microseconds" << endl ;
        // cross checking random arrays
        cout << "Random Array" << endl ;
        print_array( input ) ;
        begin_sel = clock( ) ;
        // here & operator isn't used so the sorted array won't be saved at the address we are accessing
from input[ ] in the main function
        selection_sort( input , size ) ;
        end_sel = clock( ) ;
        // calculating runtime of sorting and adding it to the sum of runtimes
        sum_sel += double( end_sel - begin_sel ) ;
        cout << "Execution Time for " << ( i + 1 ) << " time for selection sort is " << double( end_sel - begin_sel )
<< " microseconds" << endl ;
    }
    // printing average of the runtimes of insertion and selection sort
    cout << "Average runtime for all 3 sortings of insertion sort is " << ( sum_ins / 3 )  << " microseconds" <<
endl ;
    cout << "Average runtime for all 3 sortings of selection sort is " << ( sum_sel / 3 )  << " microseconds" <<
endl ;
}
```

// **Input-2 Plot-2 Main Function**

```
// main function which calls the respective algorithm for sorting.
int main()
{
    // initializing variables and arrays
    vector< int > input ;
    int size , j ;
    // asking user for input of the number of elements
    cout << "Give the number of elements you would like the randomly generated input to have." << endl ;
    cin >> size ;
    // time variables
    clock_t begin_ins , end_ins , begin_sel , end_sel ;
    // pushing size number of random values in range [ 1 , size ]
    for( j = 0 ; j < size ; ++j )
        input.push_back( rand( ) % ( size ) + 1 ) ;
    sort( input.begin( ) , input.end( ) ) ;
    // cross checking sorted array
    cout << "Non-decreasing Array" << endl ;
    print_array( input ) ;
    begin_ins = clock( ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    insertion_sort( input , size ) ;
    end_ins = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time for insertion sort is " << double( end_ins - begin_ins ) << " microseconds" << endl ;
    // cross checking random arrays
    cout << "Non-decreasing Array" << endl ;
    print_array( input ) ;
    begin_sel = clock( ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    selection_sort( input , size ) ;
    end_sel = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time for selection sort is " << double( end_sel - begin_sel ) << " microseconds" << endl
;
}
```

// **Input-3 Plot-3 Main Function**

```
// main function which calls the respective algorithm for sorting.
int main()
{
    // initializing variables and arrays
    vector< int > input ;
    int size , j ;
    // asking user for input of the number of elements
    cout << "Give the number of elements you would like the randomly generated input to have." << endl ;
    cin >> size ;
    // time variables
    clock_t begin_ins , end_ins , begin_sel , end_sel ;
    // pushing size number of random values in range [ 1 , size ]
    for( j = 0 ; j < size ; ++j )
        input.push_back( rand( ) % ( size ) + 1 ) ;
    sort( input.begin( ) , input.end( ) , greater< int >( ) ) ;
    // cross checking sorted array
    cout << "Non-increasing Array" << endl ;
    print_array( input ) ;
    begin_ins = clock( ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    insertion_sort( input , size ) ;
    end_ins = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time for insertion sort is " << double( end_ins - begin_ins ) << " microseconds" << endl ;
    // cross checking sorted array
    cout << "Non-increasing Array" << endl ;
    print_array( input ) ;
    begin_sel = clock( ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    selection_sort( input , size ) ;
    end_sel = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time selection sort is " << double( end_sel - begin_sel ) << " microseconds" << endl ;
}

// Input-4 Plot-4 Main Function
// main function which calls the respective algorithm for sorting.
int main()
{
    // initializing variables and arrays
    vector< int > input ;
    int size , j ;
    // asking user for input of the number of elements
    cout << "Give the number of elements you would like the randomly generated input to have." << endl ;
    cin >> size ;
    // initializing variables with values
    double sum_ins = 0.0 ;
    double sum_sel = 0.0 ;
    // time variables
    clock_t begin_ins , end_ins , begin_sel , end_sel ;
    for( int i = 0 ; i < 3 ; ++i ){
        // clearing input vector from the previous values stored so that a new set of random values can be
added and later sorted to make 3 random sortings distinct
        // pushing size number of random values in range [ 1 , size ]
        input.clear( ) ;
        for( j = 0 ; j < size ; ++j )
            input.push_back( rand( ) % ( size ) + 1 ) ;
        sort( input.begin( ) , input.end( ) ) ;
        for( j = 0 ; j < 50 ; ++j )
            swap( &input[ rand( ) % size ] , &input[ rand( ) % size ] ) ;
        // cross checking random arrays
        cout << "Random Array" << endl ;
        print_array( input ) ;
        begin_ins = clock( ) ;
        // here & operator isn't used so the sorted array won't be saved at the address we are accessing
from input[ ] in the main function
        insertion_sort( input , size ) ;
        end_ins = clock( ) ;
        // calculating runtime of sorting and adding it to the sum of runtimes
        sum_ins += double( end_ins - begin_ins ) ;
        cout << "Execution Time for " << ( i + 1 ) << " time for insertion sort is " << double( end_ins - begin_ins ) <<
" microseconds" << endl ;
        // cross checking random arrays
        cout << "Random Array" << endl ;
        print_array( input ) ;
        begin_sel = clock( ) ;
        // here & operator isn't used so the sorted array won't be saved at the address we are accessing
from input[ ] in the main function
```

```
        selection_sort( input , size ) ;
        end_sel = clock( ) ;
        // calculating runtime of sorting and adding it to the sum of runtimes
        sum_sel += double( end_sel - begin_sel ) ;
        cout << "Execution Time for " << ( i + 1 ) << " time for selection sort is " << double( end_sel - begin_sel )
<< " microseconds" << endl ;
    }
    // printing average of the runtimes of insertion and selection sort
    cout << "Average runtime for all 3 sortings of insertion sort is " << ( sum_ins / 3 )  << " microseconds" <<
endl ;
    cout << "Average runtime for all 3 sortings of selection sort is " << ( sum_sel / 3 )  << " microseconds" <<
endl ;
}
```

// **Input-5 Main Function**

```
// main function which calls the respective algorithm for sorting.
int main()
{
    // initializing variables and arrays
    vector< int > input ;
    int size = 100000 ;
    // pushing size number of random values in range [ 1 , size ]
    for( int i = 0 ; i < size ; ++i )
        input.push_back( rand( ) % ( 50 ) + 1 ) ;
    // time variables
    clock_t begin_ins , end_ins , begin_sel , end_sel ;
    // cross checking random arrays
    cout << "Random Array" << endl ;
    print_array( input ) ;
    begin_ins = clock( ) ;
    insertion_sort( input , size ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    end_ins = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time for insertion sort is " << double( end_ins - begin_ins ) << " microseconds" << endl ;
    // cross checking random arrays
    cout << "Random Array" << endl ;
    print_array( input ) ;
    begin_sel = clock( ) ;
    // here & operator isn't used so the sorted array won't be saved at the address we are accessing from
input[ ] in the main function
    selection_sort( input , size ) ;
    end_sel = clock( ) ;
    // calculating runtime of sorting and adding it to the sum of runtimes
    cout << "Execution Time for selection sort is " << double( end_sel - begin_sel ) << " microseconds" << endl
;
}
```