```
/* program6.cpp
 *
 * CS 121.Bolden.................Compiler version.....................Spencer Reed
 * 12/08/21 ...MacBook Air, 1.1GHz quad-core Intel Core i5...reed7385@vandals.uidaho.edu
 *
 * This program will take 9 different arrays of numbers and sort them using Selection,
 * Quick, and Merge sorting methods.
 *-------------------------------------------------------------------
 */

#include <iostream>
#include <stdio.h>
#include <time.h>
#include <cstdlib>

using namespace std;

void Selectionsort(int a[], int n);
int IndexOfSmallest(int a[], int iStart, int iEnd);
void Swap(int &i, int &iSmallest);
void Quicksort(int a[], int first, int last);
int Pivot(int a[], int first, int last);
void Mergesort(int a[], int first, int last, int n);
void Merge(int a[], int firstLeft, int lastLeft,
          int firstRight, int lastRight, int n);
void FillArrayRand(int a[], int n);
void FillArrayInOrder(int a[], int n);
void FillArrayBackwards(int a[], int n);

int ARRAY_SIZE_1 = 1000;
int ARRAY_SIZE_2 = 10000;
int ARRAY_SIZE_3 = 100000;

int swapsS = 0;
int swapsQ = 0;
int swapsM = 0;

int main()
{
       double timeUsedTotalS;
       double timeUsedTotalQ;
       double timeUsedTotalM;

       int swapsTotalS;
       int swapsTotalQ;
       int swapsTotalM;

       int arrayS1[ARRAY_SIZE_1];
       int arrayS2[ARRAY_SIZE_2];
       int arrayS3[ARRAY_SIZE_3];

       int arrayQ1[ARRAY_SIZE_1];
       int arrayQ2[ARRAY_SIZE_2];
       int arrayQ3[ARRAY_SIZE_3];

       int arrayM1[ARRAY_SIZE_1];
       int arrayM2[ARRAY_SIZE_2];
       int arrayM3[ARRAY_SIZE_3];

       FillArrayRand(arrayS1, ARRAY_SIZE_1);
       FillArrayRand(arrayQ1, ARRAY_SIZE_1);
       FillArrayRand(arrayM1, ARRAY_SIZE_1);

       FillArrayInOrder(arrayS2, ARRAY_SIZE_2);
       FillArrayInOrder(arrayQ2, ARRAY_SIZE_2);
       FillArrayInOrder(arrayM2, ARRAY_SIZE_2);
```

```
FillArrayBackwards(arrayS3, ARRAY_SIZE_3);
FillArrayBackwards(arrayQ3, ARRAY_SIZE_3);
FillArrayBackwards(arrayM3, ARRAY_SIZE_3);

cout << endl;
cout << "-----------------------------------------------------" << endl;
cout << "------------SELECTION SORTING FUNCTIONS------------" << endl;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Selection Array Contents (RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS1[i] << ", " << endl;
}
clock_t start = clock();
Selectionsort(arrayS1, ARRAY_SIZE_1);
clock_t end = clock();
double timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalS = timeUsed;
cout << endl;
cout << "First 20 Selection Sorted Array Contents (RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS1[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsS << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
swapsS = 0;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Selection Array Contents (IN ORDER): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS2[i] << ", " << endl;
}
start = clock();
Selectionsort(arrayS2, ARRAY_SIZE_2);
end = clock();
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalS = timeUsedTotalS + timeUsed;
cout << endl;
cout << "First 20 Selection Sorted Array Contents (IN ORDER): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS2[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsS << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
swapsTotalS = swapsS;
swapsS = 0;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Selection Array Contents (BACKWARDS): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS3[i] << ", " << endl;
}
start = clock();
Selectionsort(arrayS3, ARRAY_SIZE_3);
end = clock();
```

```
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalS = timeUsedTotalS + timeUsed;
cout << endl;
cout << "First 20 Selection Sorted Array Contents (BACKWARDS): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayS3[i] << ", " << endl;
}
cout << endl;
swapsTotalS = swapsTotalS + swapsS;
cout << "Swaps needed: " << swapsS << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
cout << "Selection Sort Total Swaps: " << swapsTotalS << endl;
printf("Total Time Used In Selection Sort: %lf \n", timeUsedTotalS);
cout << endl;
cout << endl;
swapsS = 0;

cout << "-----------------------------------------------------" << endl;
cout << "--------------QUICK SORTING FUNCTIONS--------------" << endl;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Quick Array Contents (RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ1[i] << ", " << endl;
}
start = clock();
Quicksort(arrayQ1, 0, 999);
end = clock();
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalQ = timeUsedTotalQ + timeUsed;
cout << endl;
cout << "First 20 Quick Sorted Array Contents(RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ1[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsQ << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
swapsTotalQ = swapsQ;
swapsQ = 0;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Quick Array Contents (IN ORDER): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ2[i] << ", " << endl;
}
start = clock();
Quicksort(arrayQ2, 0, 9999);
end = clock();
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalQ = timeUsedTotalQ + timeUsed;
cout << endl;
cout << "First 20 Quick Sorted Array Contents(IN ORDER): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ2[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsQ << endl;
```

```
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
swapsTotalQ = swapsTotalQ + swapsQ;
swapsQ = 0;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Quick Array Contents (BACKWARDS): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ3[i] << ", " << endl;
}
start = clock();
Quicksort(arrayQ3, 0, 99999);
end = clock();
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalQ = timeUsedTotalQ + timeUsed;
cout << endl;
cout << "First 20 Quick Sorted Array Contents(BACKWARDS): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayQ3[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsQ << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
swapsTotalQ = swapsTotalQ + swapsQ;
cout << endl;
cout << "Quick Sort Total Swaps: " << swapsTotalQ << endl;
printf("Total Time Used In Quick Sort: %lf \n", timeUsedTotalQ);
cout << endl;
cout << endl;
swapsQ = 0;

cout << "-----------------------------------------------------" << endl;
cout << "--------------MERGE SORTING FUNCTIONS--------------" << endl;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Merge Array Contents (RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayM1[i] << ", " << endl;
}
start = clock();
Mergesort(arrayM1, 0, 999, ARRAY_SIZE_1);
end = clock();
timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
timeUsedTotalM = timeUsedTotalM + timeUsed;
cout << endl;
cout << "First 20 Merge Sorted Array Contents (RANDOM): " << endl;
for(int i = 0; i < 20; i++)
{
	cout << arrayM1[i] << ", " << endl;
}
cout << endl;
cout << "Swaps needed: " << swapsM << endl;
printf("Sorting took %lf seconds to execute \n", timeUsed);
cout << endl;
swapsTotalM = swapsM;
swapsM = 0;
cout << "-----------------------------------------------------" << endl;
cout << endl;

cout << "Initial 20 Merge Array Contents (IN ORDER): " << endl;
for(int i = 0; i < 20; i++)
{
```

```
		cout << arrayM2[i] << ", " << endl;
	}
	start = clock();
	Mergesort(arrayM2, 0, 9999, ARRAY_SIZE_2);
	end = clock();
	timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
	timeUsedTotalM = timeUsedTotalM + timeUsed;
	cout << endl;
	cout << "First 20 Merge Sorted Array Contents (RANDOM): " << endl;
	for(int i = 0; i < 20; i++)
	{
		cout << arrayM2[i] << ", " << endl;
	}
	cout << endl;
	cout << "Swaps needed: " << swapsM << endl;
	printf("Sorting took %lf seconds to execute \n", timeUsed);
	cout << endl;
	swapsTotalM = swapsTotalM + swapsM;
	swapsM = 0;
	cout << "-----------------------------------------------------" << endl;
	cout << endl;

	cout << "Initial 20 Merge Array Contents (BACKWARDS): " << endl;
	for(int i = 0; i < 20; i++)
	{
		cout << arrayM3[i] << ", " << endl;
	}
	start = clock();
	Mergesort(arrayM3, 0, 99999, ARRAY_SIZE_3);
	end = clock();
	timeUsed = ((double)(end - start)) / CLOCKS_PER_SEC;
	timeUsedTotalM = timeUsedTotalM + timeUsed;
	cout << endl;
	cout << "First 20 Merge Sorted Array Contents (BACKWARDS): " << endl;
	for(int i = 0; i < 20; i++)
	{
		cout << arrayM3[i] << ", " << endl;
	}
	cout << endl;
	swapsTotalM = swapsTotalM + swapsM;
	cout << "Swaps needed: " << swapsM << endl;
	printf("Sorting took %lf seconds to execute \n", timeUsed);
	cout << endl;
	cout << "Merge Sort Total Swaps: " << swapsTotalM << endl;
	printf("Total Time Used In Merge Sort: %lf \n", timeUsedTotalM);
	return 0;
}

void FillArrayRand(int a[], int n)
{
	for(int i = 0; i < n; i++)
	{
		a[i] = rand() % n + 1;
	}
}

void FillArrayInOrder(int a[], int n)
{
	for(int i = 0; i < n; i++)
	{
		a[i] = i+1;
	}
}

void FillArrayBackwards(int a[], int n)
{
	for(int i = 0; i < n; i++)
```

```
	{
		a[i] = n - i;
	}
}

void Selectionsort(int a[], int n)
{
	int iSmallest;

	for(int i = 0; i<n-1; i++)
	{
		iSmallest = i;
		for(int j = i+1; j<n; j++)
		{
			  if(a[j] < a[iSmallest])
			{
				iSmallest = j;
			}
		}
		Swap(a[i], a[iSmallest]);
		swapsS++;
	}
}


void Swap(int &i, int &iSmallest)
{
	int temp = i;
	i = iSmallest;
	iSmallest = temp;
}

void Quicksort(int a[], int first, int last)
{
	int pivot;

	if(first < last)
	{
		pivot = Pivot(a, first, last);
		Quicksort(a, first, pivot-1);
		Quicksort(a, pivot+1, last);
		swapsQ++;
	}
}

int Pivot(int a[], int first, int last)
{
	int p = first;
	int pivot = a[first];

	for(int i = first+1; i <= last; i++)
	{
		if(a[i] <= pivot)
		{
			p++;
			Swap(a[i], a[p]);
		}
	}
	Swap(a[p], a[first]);
	return p;
}

void Mergesort(int a[], int first, int last, int n)
{
	int middle;

	if(first < last)
```

```
	{
		middle = (first + last) / 2;
		Mergesort(a, first, middle, n);
		Mergesort(a, middle+1, last, n);
		Merge(a, first, middle, middle+1, last, n);
	}
}

void Merge(int a[], int firstLeft, int lastLeft,
	    int firstRight, int lastRight, int n)
{
	swapsM++;
	int temp[n];
	int index = firstLeft;
	int firstSave = firstLeft;

	while((firstLeft <= lastLeft) && (firstRight <= lastRight))
	{
		if(a[firstLeft] < a[firstRight])
		{
			temp[index] = a[firstLeft];
			firstLeft++;
		}
		else
		{
			temp[index] = a[firstRight];
			firstRight++;
		}
		index++;
	}

	while(firstLeft <= lastLeft)
	{
		temp[index++] = a[firstLeft];
		firstLeft++;
	}

	while(firstRight <= lastRight)
	{
		temp[index++] = a[firstRight];
		firstRight++;
	}

	for(index = firstSave; index <= lastRight; index++)
		a[index] = temp[index];
}
```