Selection and Insertion Sort Performance Evaluation

Objective

The purpose of this exercise is to experimentally assess the efficiency of **Selection Sort** and **Insertion Sort**. You will generate random permutations of arrays of numbers from 0 to $N - 1$, sort them, and record the time taken. To improve measurement accuracy, average timing results over multiple runs.

Tasks

1. Implement Selection Sort and Insertion Sort algorithms in C++.
2. Create random permutations of size N, and sort them over `RUNS` iterations.
3. Measure and report average time taken (t_{avg}) and compute t_{avg}/N^2.
4. Plot:
   - N vs t_{avg} (should resemble a parabola).
   - N vs t_{avg}/N^2 to estimate the asymptotic constant c.
5. Predict runtime for large N using the equation: $t_{avg} = c \times N^2$.

Experimental Protocol

- Begin with $N = 1000$, `RUNS` $= 1024$
- When total runtime exceeds 60 seconds, halve `RUNS`
- Continue until `RUNS` $= 1$ and runtime exceeds 60s
- Use results to fill a table like the one below:

N	t_{avg} (sec)	# RUNS	N^2	t_{avg}/N^2
1000	...	1024	...	...
2000	...	1024	...	...
...	...	...	...	...

Code Snippets (Header Only)

sorting.h

```
#ifndef SORTING_H
#define SORTING_H
#include <vector>
std::vector<int> permutation(int n);
void selectionSort(std::vector<int>& arr);
void insertionSort(std::vector<int>& arr);
#endif
```

stopwatch.h (uses chrono)

```
#ifndef STOPWATCH_H
#define STOPWATCH_H
#include <chrono>
std::chrono::steady_clock::time_point start;
inline void startTimer() {
  start = std::chrono::steady_clock::now();
}
inline double stopTimer() {
  auto end = std::chrono::steady_clock::now();
  std::chrono::duration<double> elapsed = end - start;
  return elapsed.count();
}
#endif
```

Permutation

```
// Generate a random permutation of integers [0, N-1]
// mt19937 is a Mersenne Twister random number generator
vector<int> generatePermutation(int n) {
    vector<int> arr(n);
    for (int i = 0; i < n; ++i) arr[i] = i;
    shuffle(arr.begin(), arr.end(), mt19937(random_device()()));
    return arr;
}
```

Deliverables

1. A table summarizing N, t_{avg}, N^2, t_{avg}/N^2 for each algorithm
2. Two plots:
   - N vs t_{avg}
   - N vs t_{avg}/N^2
3. Predictions of runtime for $N = 10^5, 10^6, 10^7, ...$ based on the asymptotic constant
4. Discussion: compare the performance and plot shapes for Selection and Insertion Sort

Optional Extension

Modify your program to count and record the number of swaps or shifts performed for $N = 1000$, `RUNS` $= 1000$ and plot a histogram of the swap counts. Comment on the spread with respect to N^2.