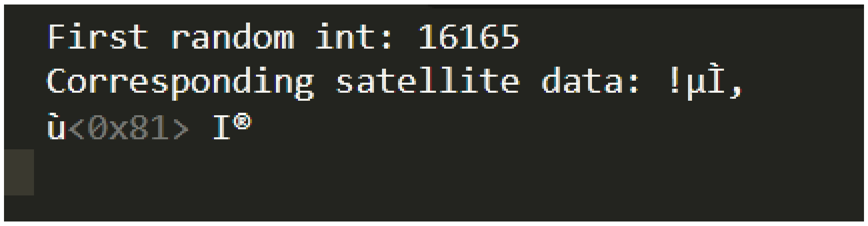
Vaghamashi Kishan
202312014

1)

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;

string generateSatelliteData() {
    string data;
    for (int i = 0; i < 10; ++i) {
        data.push_back(static_cast<int>(rand() % 256));
    }
    return data;
}

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));
    vector<int> randomIntegers; vector<pair<int, string>> satelliteData;
    for (int i = 0; i < 10000; ++i) {
        randomIntegers.push_back(rand());
    }

    for (const auto& integer : randomIntegers) {
        satelliteData.push_back(make_pair(integer,
generateSatelliteData()));
        cout << "First random int: " << satelliteData[0].first << endl;
        cout << "Corresponding satellite data: " << satelliteData[0].second
<< endl;
        return 0;
    }
}
```

Output:



```
First random int: 16165
Corresponding satellite data: !µÌ,
ù<0x81> I®
```

Vaghamashi Kishan
202312014

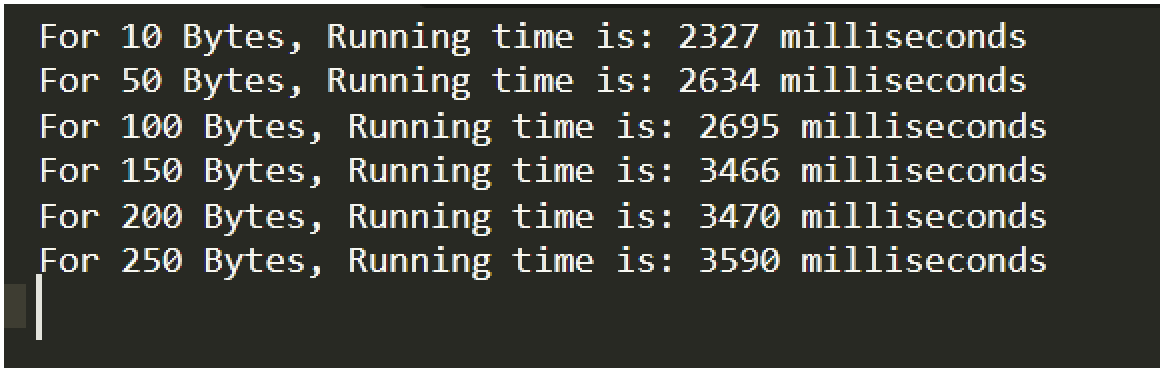
2)

```
#include <iostream>
#include <vector>
#include <chrono>
#include <cstdlib>
#include <ctime>
using namespace std;
void insertionSort(vector<pair<int, string>> &arr)
{
    int n = arr.size();
    for (int i = 1; i < n; ++i)
    {
        auto key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j].second > key.second)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

string generateSatelliteData(int n)
{
    string satelliteData;
    for (int i = 0; i < n; ++i)
    {
        satelliteData.push_back(static_cast<int>(rand() % 256));
    }
    return satelliteData;
}

int main()
{
    int n[6] = {10, 50, 100, 150, 200, 250};
    for (int i = 0; i < 6; ++i)
    {
        srand(static_cast<unsigned int>(time(nullptr)));
        vector<int> randomIntegers;
        vector<pair<int, string>> randomIntegersWithSatellite;
        for (int i = 0; i < 10000; ++i)
        {
            randomIntegers.push_back(rand());
```

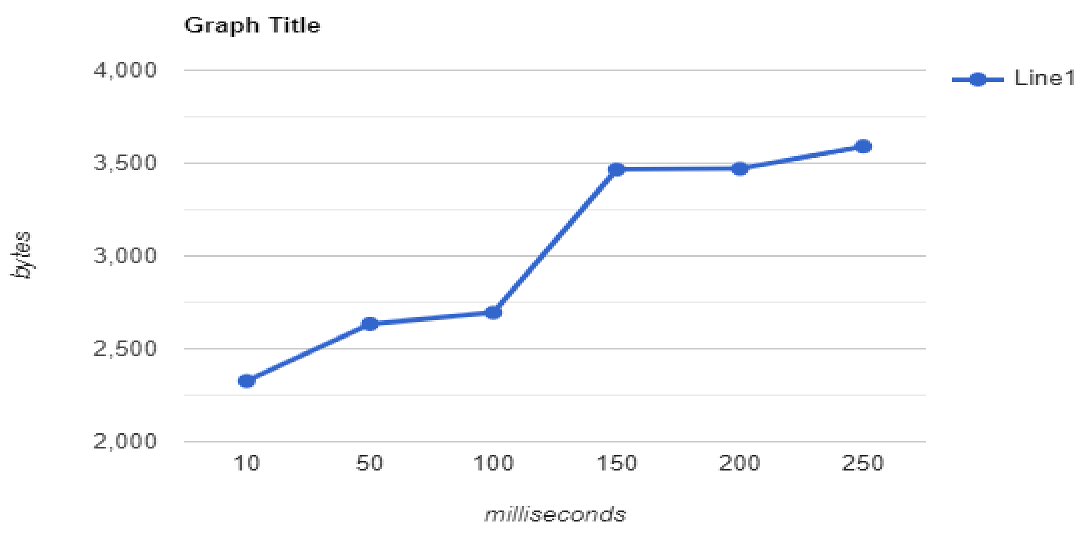
Vaghamashi Kishan
202312014

```
        }
        for (const auto &integer : randomIntegers)
        {
            randomIntegersWithSatellite.push_back(make_pair(integer,
generateSatelliteData(n[i])));
        }
        auto start = chrono::steady_clock::now();
        insertionSort(randomIntegersWithSatellite);
        auto end = chrono::steady_clock::now();
        auto duration = chrono::duration_cast<chrono::milliseconds>(end -
start);
        cout << "For " << n[i] << " Bytes, Running time is: " <<
duration.count() << " milliseconds" << endl;
    }
    return 0;
}
```

Output:



```
For 10 Bytes, Running time is: 2327 milliseconds
For 50 Bytes, Running time is: 2634 milliseconds
For 100 Bytes, Running time is: 2695 milliseconds
For 150 Bytes, Running time is: 3466 milliseconds
For 200 Bytes, Running time is: 3470 milliseconds
For 250 Bytes, Running time is: 3590 milliseconds
```



Vaghamashi Kishan
202312014

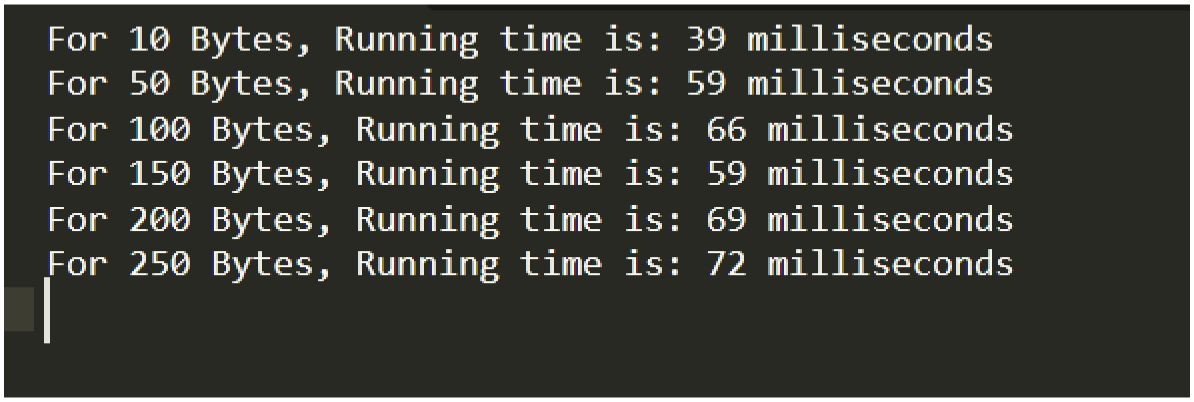
3)

```cpp
#include <iostream>
#include <vector>
#include <chrono>
#include <cstdlib>
#include <ctime>
using namespace std;
void merge(vector<pair<int, string>> &arr, int l, int m, int r)
{
    int n1 = m - l + 1;
    int n2 = r - m;
    vector<pair<int, string>> L(n1), R(n2);
    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    int i = 0;
    int j = 0;
    int k = l;
    while (i < n1 && j < n2)
    {
        if (L[i].second <= R[j].second)
        {
            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++;
```

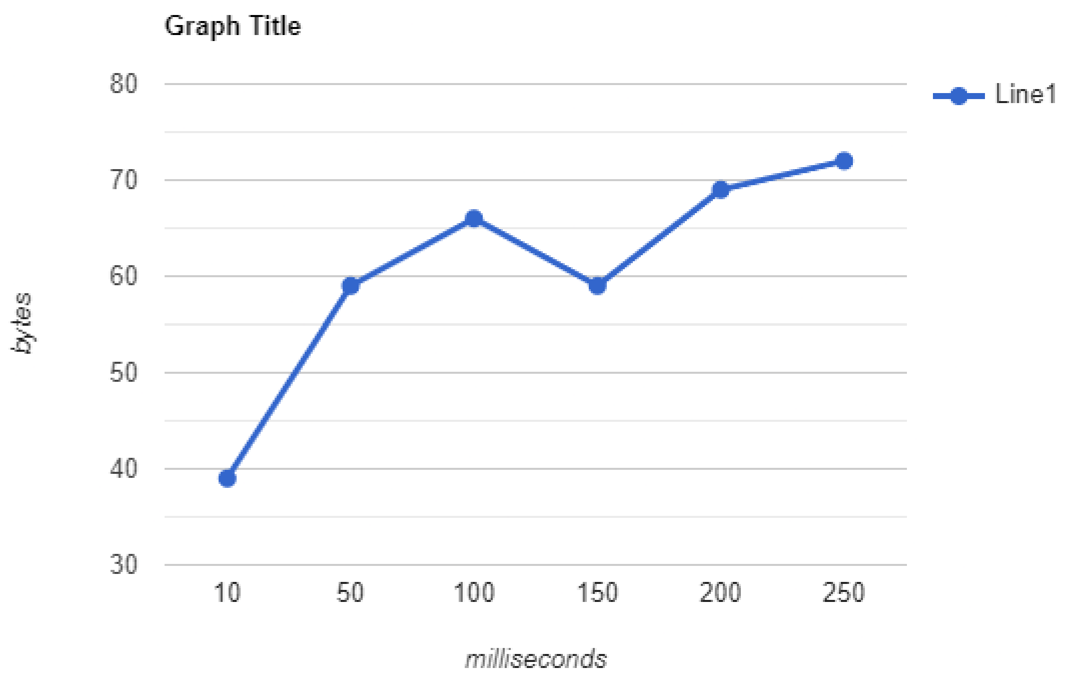
Vaghamashi Kishan
202312014

```cpp
    }
}

void mergeSort(vector<pair<int, string>> &arr, int l, int r)
{
    if (l < r)
    {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
string generateSatelliteData(int n)
{
    string satelliteData;
    for (int i = 0; i < n; ++i)
    {
        satelliteData.push_back(static_cast<int>(rand() % 256));
    }
    return satelliteData;
}

int main()
{
    int n[6] = {10, 50, 100, 150, 200, 250};
    for (int i = 0; i < 6; ++i)
    {
        srand(static_cast<unsigned int>(time(nullptr)));
        vector<pair<int, string>> satelite;
        for (int j = 0; j < 10000; ++j)
        {
            int randomInteger = rand();
            string satelliteData = generateSatelliteData(n[i]);
            satelite.push_back(make_pair(randomInteger,
                                             satelliteData));
        }
        auto start = chrono::steady_clock::now();
        mergeSort(satelite, 0, satelite.size() - 1);
        auto end = chrono::steady_clock::now();
        auto duration = chrono::duration_cast<chrono::milliseconds>(end -
start);
        cout << "For " << n[i] << " Bytes, Running time is: " <<
duration.count() << " milliseconds" << endl;
    }
    return 0;
}
```

Output:



```
For 10 Bytes, Running time is: 39 milliseconds
For 50 Bytes, Running time is: 59 milliseconds
For 100 Bytes, Running time is: 66 milliseconds
For 150 Bytes, Running time is: 59 milliseconds
For 200 Bytes, Running time is: 69 milliseconds
For 250 Bytes, Running time is: 72 milliseconds
```



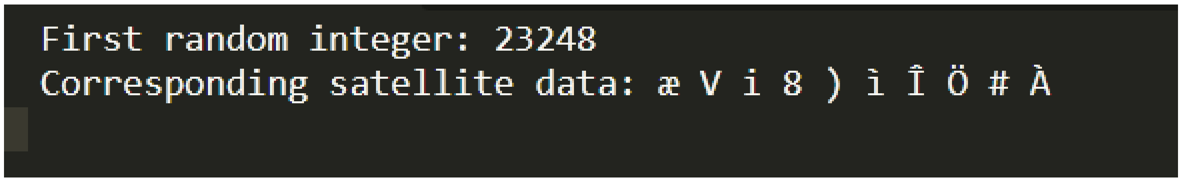
Vaghamashi Kishan
202312014

4)

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;
char *generateSatelliteData()
{
    char *temp = new char[10];
    for (int i = 0; i < 10; ++i)
    {
        temp[i] = rand() % 256;
    }
    return temp;
}
int main()
{
    srand(static_cast<unsigned int>(time(nullptr)));
    vector<int> randomIntegers;
    vector<pair<int, char *>> satelliteData;
    for (int i = 0; i < 10000; ++i)
    {
        randomIntegers.push_back(rand());
    }
    for (const auto &integer : randomIntegers)
    {
        satelliteData.push_back(make_pair(integer,
generateSatelliteData()));
    }
    cout << "First random integer: " << satelliteData[0].first << endl;
    cout << "Corresponding satellite data: ";
    for (int i = 0; i < 10; ++i)
    {
        cout << satelliteData[0].second[i] << " ";
    }
    cout << endl;
    for (auto &data : satelliteData)
    {
        delete[] data.second;
    }
    return 0;
}
```

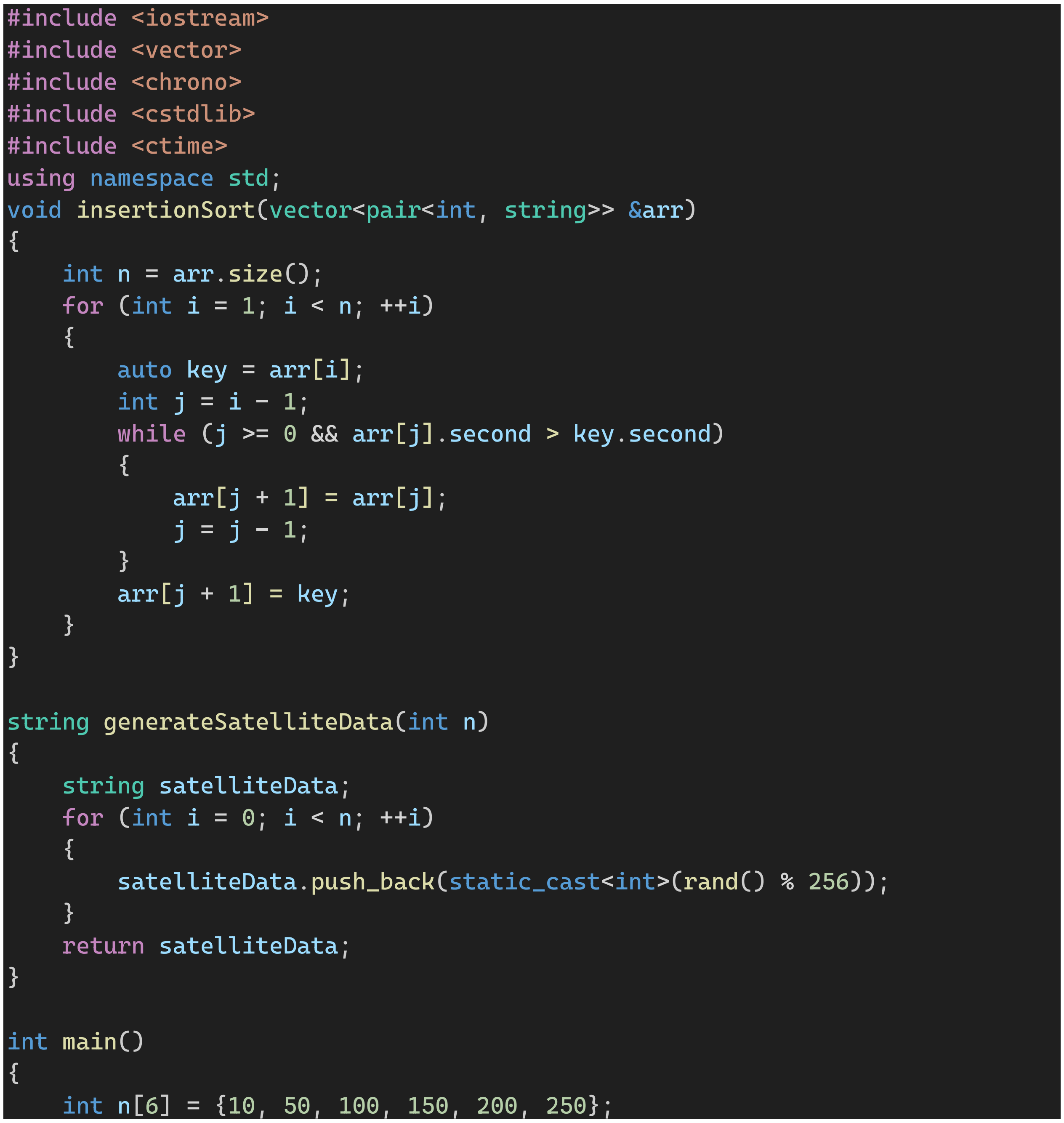
Vaghamashi Kishan
202312014

Output:



```
First random integer: 23248
Corresponding satellite data: æ V i 8 ) ì Î Ö # À
```

5)

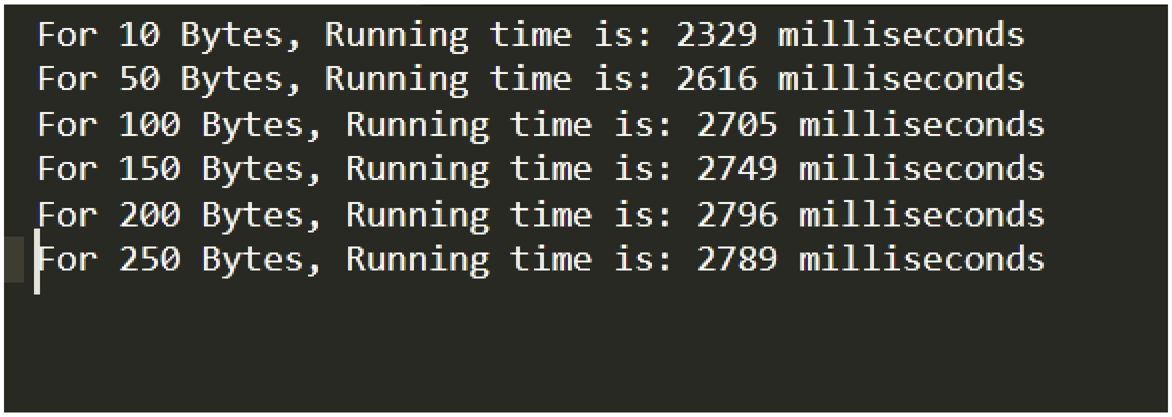


```cpp
#include <iostream>
#include <vector>
#include <chrono>
#include <cstdlib>
#include <ctime>
using namespace std;
void insertionSort(vector<pair<int, string>> &arr)
{
    int n = arr.size();
    for (int i = 1; i < n; ++i)
    {
        auto key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j].second > key.second)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

string generateSatelliteData(int n)
{
    string satelliteData;
    for (int i = 0; i < n; ++i)
    {
        satelliteData.push_back(static_cast<int>(rand() % 256));
    }
    return satelliteData;
}

int main()
{
    int n[6] = {10, 50, 100, 150, 200, 250};
```

Vaghamashi Kishan
202312014

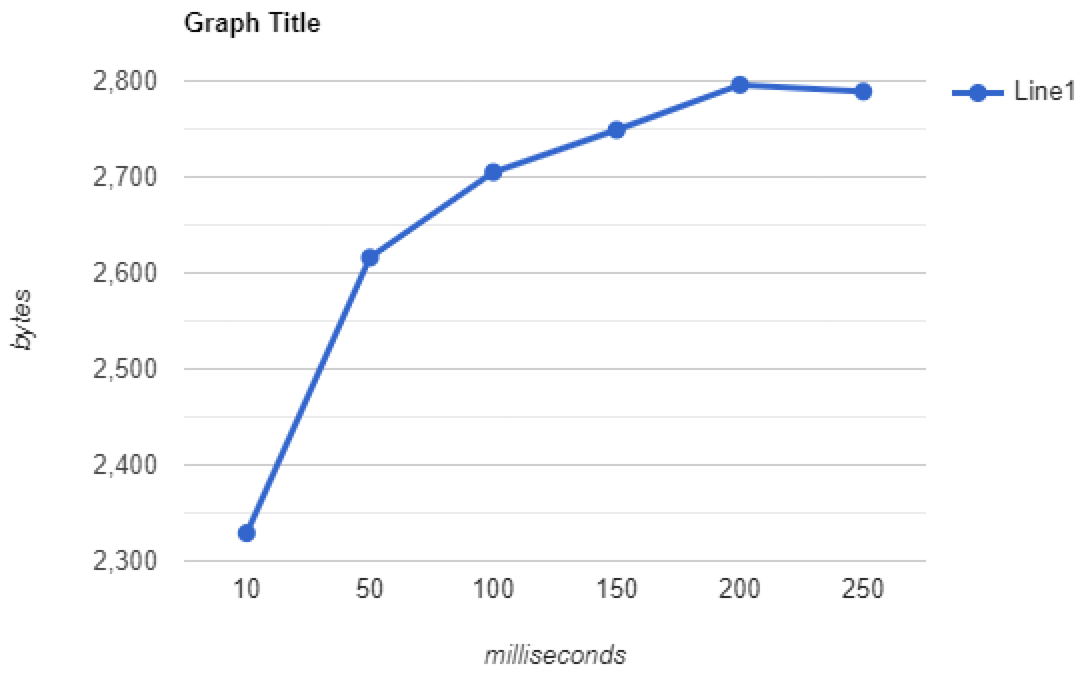
```
    for (int i = 0; i < 6; ++i)
    {
        srand(static_cast<unsigned int>(time(nullptr)));
        vector<int> randomIntegers;
        vector<pair<int, string>> randomIntegersWithSatellite;
        for (int i = 0; i < 10000; ++i)
        {
            randomIntegers.push_back(rand());
        }
        for (const auto &integer : randomIntegers)
        {
            randomIntegersWithSatellite.push_back(make_pair(integer,
                                                            generateSatellit
eData(n[i])));
        }
        auto start = chrono::steady_clock::now();
        insertionSort(randomIntegersWithSatellite);
        auto end = chrono::steady_clock::now();
        auto duration = chrono::duration_cast<chrono::milliseconds>(end -
start);
        cout << "For " << n[i] << " Bytes, Running time is: " <<
duration.count() << " milliseconds" << endl;
    }
    return 0;
}
```

Output:

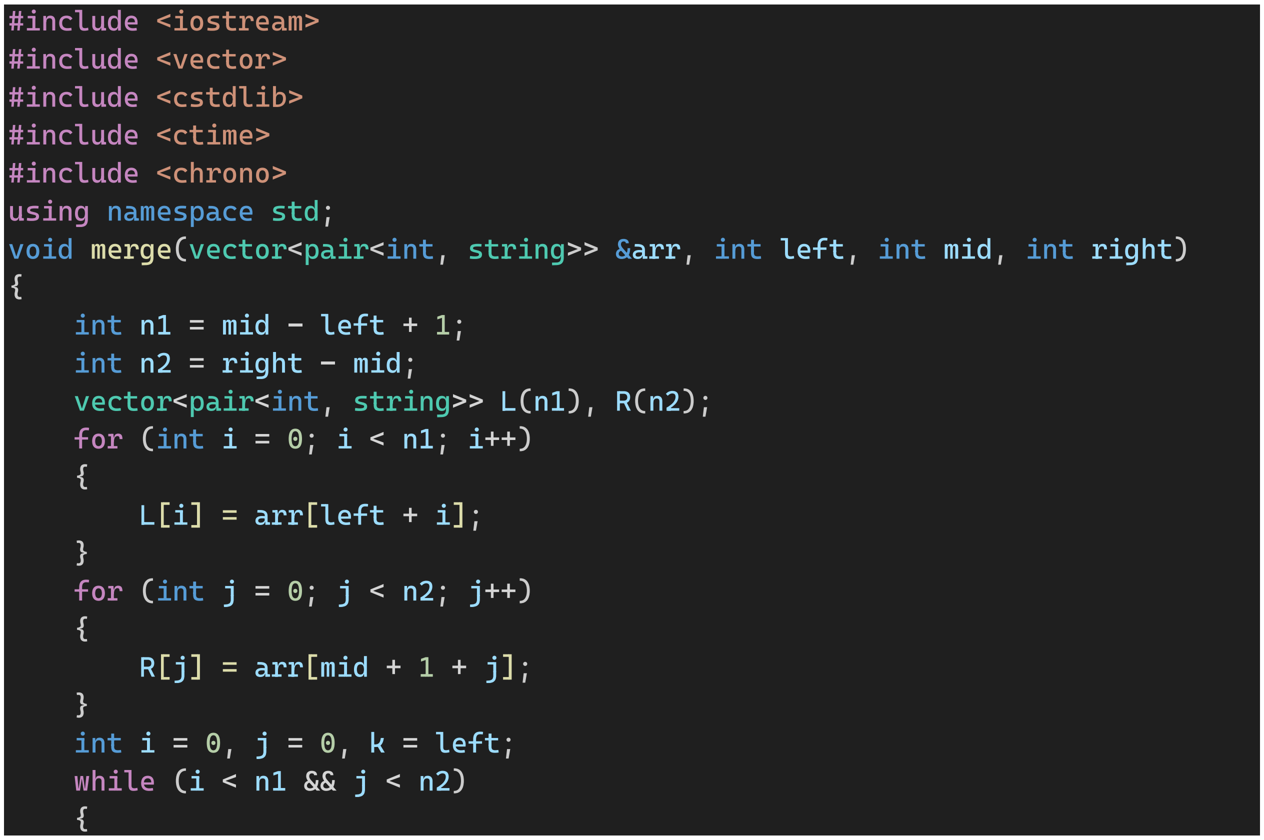


```
For 10 Bytes, Running time is: 2329 milliseconds
For 50 Bytes, Running time is: 2616 milliseconds
For 100 Bytes, Running time is: 2705 milliseconds
For 150 Bytes, Running time is: 2749 milliseconds
For 200 Bytes, Running time is: 2796 milliseconds
For 250 Bytes, Running time is: 2789 milliseconds
```

Vaghamashi Kishan
202312014



6)



```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <chrono>
using namespace std;
void merge(vector<pair<int, string>> &arr, int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;
    vector<pair<int, string>> L(n1), R(n2);
    for (int i = 0; i < n1; i++)
    {
        L[i] = arr[left + i];
    }
    for (int j = 0; j < n2; j++)
    {
        R[j] = arr[mid + 1 + j];
    }
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2)
    {
```

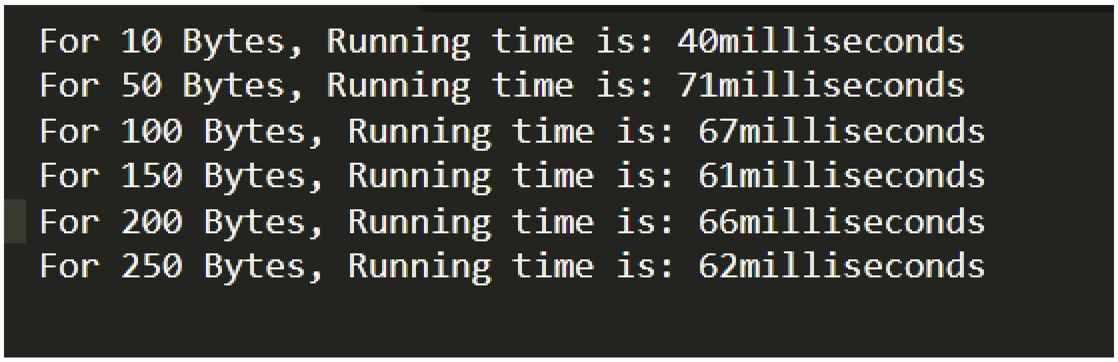
Vaghamashi Kishan
202312014

```
        if (L[i].second <= R[j].second)
        {
            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 mergeSort(vector<pair<int, string>> &arr, int left, int right)
{
    if (left < right)
    {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
string generateSatelliteData(int Bytes)
{
    string data;
    for (int i = 0; i < Bytes; ++i)
    {
        data.push_back(static_cast<char>(rand() % 256));
    }
    return data;
}
```

Vaghamashi Kishan
202312014

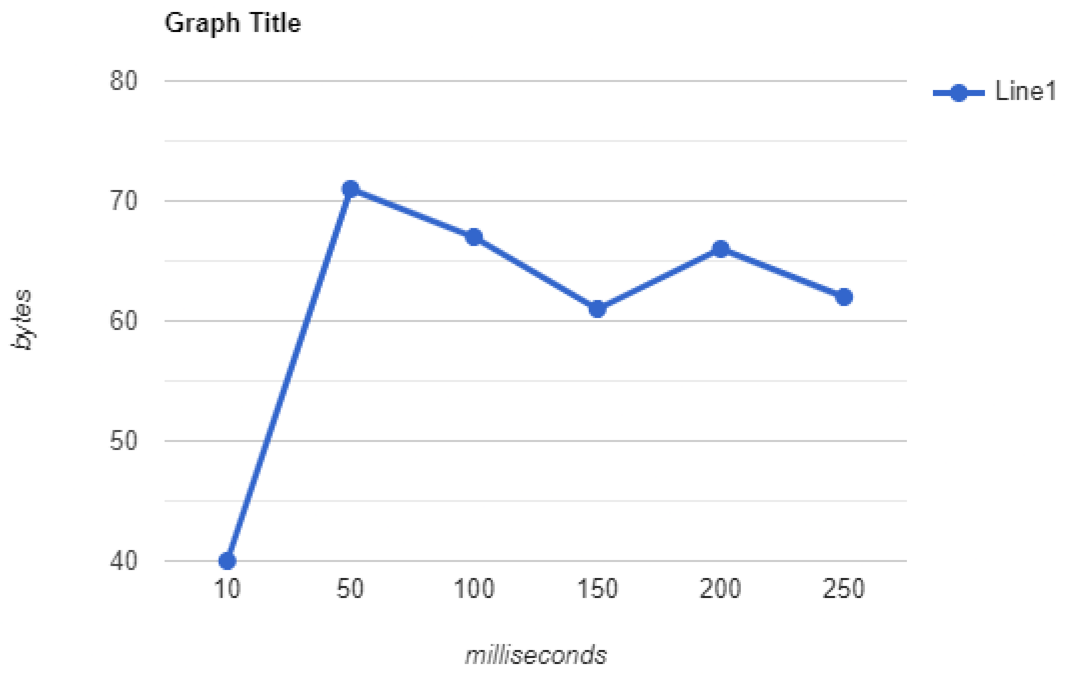
```
int main()
{
    int arr[6] = {10, 50, 100, 150, 200, 250};
    for (int i = 0; i < 6; i++)
    {
        srand(static_cast<unsigned int>(time(nullptr)));
        vector<int> randomIntegers;
        vector<pair<int, string>> satelliteData;
        for (int j = 0; j < 10000; ++j)
        {
            randomIntegers.push_back(j);
        }
        for (const auto &integer : randomIntegers)
        {
            string data = generateSatelliteData(arr[i]);
            satelliteData.push_back(make_pair(integer, data));
        }
        auto start = chrono::steady_clock::now();
        mergeSort(satelliteData, 0, satelliteData.size() - 1);
        auto end = chrono::steady_clock::now();
        auto duration = chrono::duration_cast<chrono::milliseconds>(end -
start);
        cout << "For " << arr[i] << " Bytes, Running time is: " <<
duration.count() << "milliseconds" << endl;
    }
    return 0;
}
```

Output:



```
For 10 Bytes, Running time is: 40milliseconds
For 50 Bytes, Running time is: 71milliseconds
For 100 Bytes, Running time is: 67milliseconds
For 150 Bytes, Running time is: 61milliseconds
For 200 Bytes, Running time is: 66milliseconds
For 250 Bytes, Running time is: 62milliseconds
```

Vaghamashi Kishan
202312014


Graph Title
bytes
milliseconds
Line1