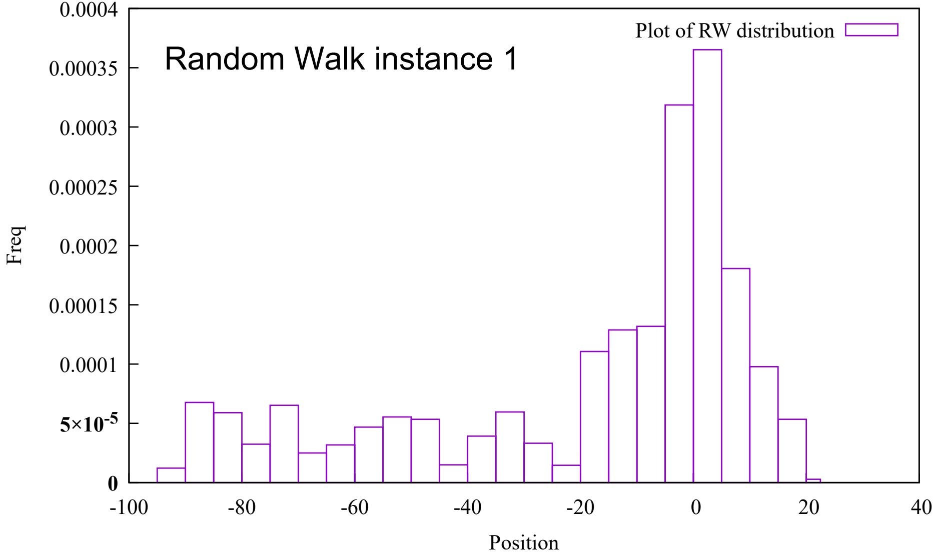
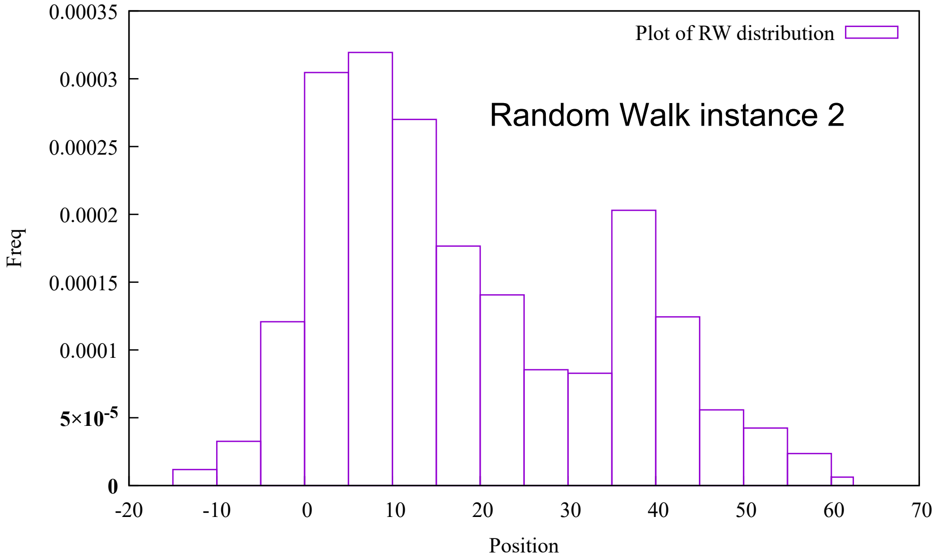
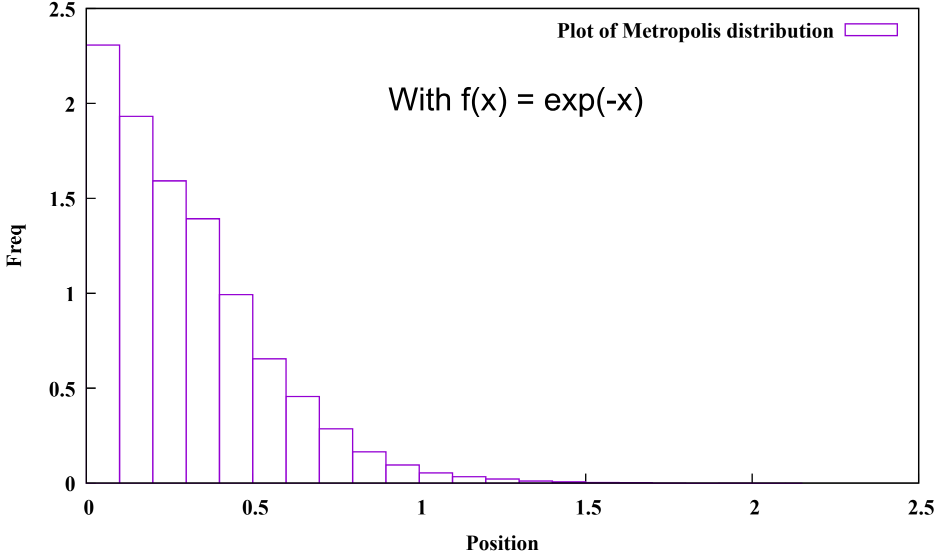
Rajat Kumar Mandal - 226121014

```cpp
#include <array>
#include <random>
#include <fstream>

//The sample size for plotting final distribution - this many numbers will be drawn
constexpr size_t samplesize = 10000;

int main() {
    std::random_device dev; //Responsible for getting a random seed from OS
    std::mt19937_64 randomwalk(350);    //Mersenne Twister engine with the seed for generating pseudo-random numbers
    std::uniform_real_distribution<double> dist(-1, 1); // distribution in range [-1, 1]

    std::array<double, samplesize> positions = {};
    double position = 0;

    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("./output/random walk.txt", std::ios::out | std::ios::trunc);

    for (auto& x : positions) {   //loop over number of samples to be drawn
        position += dist(randomwalk);
        x = position;
        outfile << x << std::endl; //write to the output file
    }

    outfile.close();    //when done, close the file.
}
```


Random Walk instance 1
Plot of RW distribution
Freq
Position
0.0004
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
5×10-5
0
-100
-80
-60
-40
-20
0
20
40


Plot of RW distribution
Random Walk instance 2
Freq
Position
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
5×10-5
0
-20
-10
0
10
20
30
40
50
60
70

```cpp
#include <array>
#include <random>
#include <fstream>

//The sample size for plotting final distribution - this many numbers will be drawn
constexpr size_t samplesize = 100000;

double f(double x){
    if (x < 0){
        return 0;
    }else{
        return exp( - x );
    }
}

int main() {
    std::random_device dev; //Responsible for getting a random seed from OS
    std::mt19937_64 randomwalk(350);    //Mersenne Twister engine with the seed for generating pseudo-random numbers
    std::mt19937_64 selector(350);    //Mersenne Twister engine with the seed for generating pseudo-random numbers
    std::uniform_real_distribution<double> randomwalkdist(-1, 1); // distribution in range [-1, 1]
    std::uniform_real_distribution<double> selectordist(0, 1); // distribution in range [0, 1]

    std::array<double, samplesize> positions = {};
    double position = 0;

    std::ofstream outfile;  //file handle to save the results in a file
    outfile.open("./output/metropolis exp.txt", std::ios::out | std::ios::trunc);

    for (auto& x : positions) {   //loop over number of samples to be drawn
        double proposed_position = position + randomwalkdist(randomwalk);

        if(selectordist(selector) <= std::min(1.0, f(proposed_position) / f(position))){
            position = proposed_position;
        }

        x = position;

        outfile << x << std::endl; //write to the output file
    }

    outfile.close();    //when done, close the file.
}
```


Plot of Metropolis distribution
With f(x) = exp(-x)
Freq
Position
0
0.5
1
1.5
2
2.5