...hysics\PH707\07 Monte Carlo Integration\problem 1.cpp 1

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

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

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

    double Sn = 0;  //counter for points inside circle

    for (size_t i = 0; i < samplesize; i++) {   //loop over number of samples to be drawn
        double x = dist(rng), y = dist(rng);    //random (x,y) coordinates
        if (x * x + y * y < 1) {    //check if inside circle, in first quadrant
            Sn++;   //increase counter
        }
    }

    std::cout << "Area of circle with radius 1: " << 4.0 * Sn / samplesize << std::endl;
}
```

The output is 3.1412

If we follow the algorithm given in the whatsapp chat by you, for problem 2,

$$S_1 = \sum_{i=0}^{10} n_i x_i$$

and

$$S_2 = \sum_{i=0}^{10} n_i f\left(x_i\right) x_i.$$

Since the samples are uniform, all n_i are equal, so we can take it outside the summation.

$$\frac{S_2}{S_1} = \frac{\sum_{i=0}^{10} n_i f\left(x_i\right) x_i}{\sum_{i=0}^{10} n_i x_i} = \frac{\sum_{i=0}^{10} f\left(x_i\right) x_i}{\sum_{i=0}^{10} x_i}.$$

Which makes no sense and is approximately 0.606024 (computated in Mathematica). However, if instead of summing the values we only sum the frequencies, we do indeed get the integral

$$\frac{\sum_{i=0}^{10} n_i f\left(x_i\right)}{\sum_{i=0}^{10} n_i} = \frac{\sum_{i=0}^{10} f\left(x_i\right)}{\sum_{i=0}^{10} 1} = \frac{1}{N} \sum_{i=0}^{10} f\left(x_i\right)$$

Which is the simpliest integration method already done in the lab before. So for this problem I follow the method of the first problem to get the correct result.

```cpp
#include <iostream>
#include <cmath>
#include <random>

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

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

    double Sn = 0;  //Counter for total

    for (size_t i = 0; i < samplesize; i++) {
        double x = dist(rng), y = dist(rng);
        if (y < exp(- x * x)) {
            Sn++;
        }
    }

    std::cout << "The integration value is: " << Sn / samplesize << std::endl;
}
```

The output is 0.74702