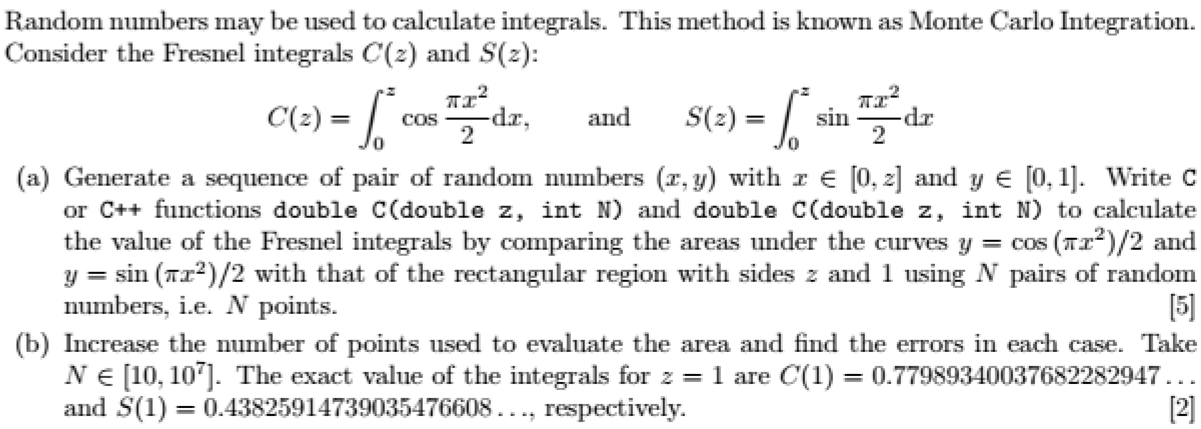

Random numbers may be used to calculate integrals. This method is known as Monte Carlo Integration. Consider the Fresnel integrals $C(z)$ and $S(z)$:

$$C(z) = \int_0^z \cos \frac{\pi x^2}{2} \mathrm{d}x, \qquad \text{and} \qquad S(z) = \int_0^z \sin \frac{\pi x^2}{2} \mathrm{d}x$$

(a) Generate a sequence of pair of random numbers (x, y) with $x \in [0, z]$ and $y \in [0, 1]$. Write `C` or `C++` functions `double C(double z, int N)` and `double C(double z, int N)` to calculate the value of the Fresnel integrals by comparing the areas under the curves $y = \cos(\pi x^2)/2$ and $y = \sin(\pi x^2)/2$ with that of the rectangular region with sides z and 1 using N pairs of random numbers, i.e. N points. [5]

(b) Increase the number of points used to evaluate the area and find the errors in each case. Take $N \in [10, 10^7]$. The exact value of the integrals for $z = 1$ are $C(1) = 0.77989340037682282947\ldots$ and $S(1) = 0.43825914739035476608\ldots$, respectively. [2]

Code:

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <fstream>
#include <climits>


using namespace std;


double f(double x)
{
  return cos(M_PI * x*x/2);
}


double g(double x)
{
  return sin(M_PI*x*x/2);
```

```
}

double C(double z, int N)
{
  int c = 0;
  double xi = 0, xf = z;
  double yi = 0, yf = 1;
  double x, y;

  for(int i=0; i<N; i++) {
    x = xi + (xf - xi) * rand()/RAND_MAX;
    y = yi + (yf - yi) * rand()/RAND_MAX;

    if(y < f(x)) c++;
  }

  return z * (double)c/(double)N;
}

double S(double z, int N)
{
  int c = 0;
  double xi = 0, xf = z;
  double yi = 0, yf = 1;
  double x, y;
  for(int i=0; i<N; i++) {
    x = xi + (xf - xi) * rand()/RAND_MAX;
    y = yi + (yf - yi) * rand()/RAND_MAX;

    if(y < g(x)) c++;
  }

  return z * (double)c/(double)N;
}

int main()
{
  double c1 = 0.779893400;
  double s1 = 0.438259147;
  double err1, err2;

  cout << INT_MAX << endl;

  ofstream fout("1.txt");

  for(int N=1; N<=7; N++) {
    err1 = abs(c1 - C(1, (int)pow(10, N)));
    err2 = abs(s1 - S(1, (int)pow(10, N)));
    fout << N << "  " << err1 << "  " << err2 << endl;
    cout << N << "  " << err1 << "  " << err2 << endl;
  }

  cout << "Done!" << endl;

  return 0;
}
```