Report on solution and implementation of Track-13 task

1) Problem description

The task requires to implement a function in C++, which for a given array A of size N with double-precision floating-point numbers and some positive integer number K, performs summation of K randomly selected elements from the array A with the uniform probability with replacement. The program must output a correct result and to be as optimised as possible. It is assumed that N and/or K can be big enough, for example, N >= 50,000,000, K >= 50,000,000.

2) Approach explanation

The program applies linear congruential method for generating pseudo-random as it is the most simple and commonly-used one.

As array length is assumed to be big enough, using default rand() may not guarantee the uniform probability of the picked array elements. That is why it was selected to write a custom rand function.

The implemented algorithm cane be described in 3 steps:

1. Seed is the value that rand function returns. It's initial value is picked as time(0) for more efficient random number generation and the sequence of random numbers is counted using $X_{n+1} = a * X_n + c$ formula. Now, for the period of sequence to be as long as possible 3 conditions should be met:

1.1) $gcd(c, N) = 1$;
1.2) Let p be the product of all prime divisors of N (1 if none is present). Then $a - 1 \equiv 0 (mod p)$;
1.3) If $N \equiv 0(mod 4)$, then $a - 1 \equiv 0(mod 4)$.

P value can easily be computed, using $O(\sqrt{N})$ algorithm which goes through every prime divisor of N and values a and can be calculated as 2 * p + 1 and 2 * p - 1 respectively, so that they will satisfy the condition above.

2. Second step is to count new random number with mentioned formula and to return it.

3. Eventually we use our custom rand function to pick K random indices and summarize all array elements corresponding to such indices. Then we return the result.

3) Code listing

The work involved using the C++ programming language. The Visual Studio environment was used to develop and test the program.

The listing below shows a software implementation of the algorithm for summarizing K random elements of array A with length N:

```
#define ll long long
#include <ctime>
#include <iostream>
using namespace std;


// Disclaimer: the following code uses linear congruential method
   for generating random numbers

// coefficient used for generation of random numbers
ll par;

// variable that stores random values
ll seed = time(0);

// function setting "par" variable, depending on the array length (
   for more efficient generation)
void par_set(ll n){
        ll b = 1;
        ll k = 2;
        while (k * k <= n){
                if (n % k == 0){
                        if (b % k != 0){
                                b *= k;
                        }
                        n /= k;
                }
                else{
                        k++;
                }
        }
        if (n > 1 and b % n != 0){
                b *= n;
        }
        par = 2 * b + 1;
}


// function generating random numbers from 0 to n
ll new_rand(ll n){
```

```cpp

                par_set(n);

                seed = (seed * par + par - 2) % n;
                return seed;
        }


        // function performing summation of k random array values
        double ran_sum(double* a, ll n, ll k){
                double s = 0;
                for (int i = 0; i < k; i++){
                        ll ind = new_rand(n);
                        s += a[ind];
                }
                return s;
        }








        int main()
        {
                // testing the implemented fuction

                ll n, k;
                cin >> n >> k;

                double* a = new double[n];
                for (int i = 0; i < n; i++){
                        cin >> a[i];
                }

                cout << ran_sum(a, n, k) << endl;


                return 0;
        }
```