// Solution 4: Storing Investment Account Balances in 1-d array
// using run-time array allocation

#include <iostream> // cin, cout
#include <cstdlib> // exit()
#include <iomanip> // setw(), setprecision()

using namespace std;

void get_maximum_investments(int& max_investments);
double* allocate_array(int max_investments);
void get_investments(double* pinv, int max_investments, int& investment_count);
void show_investments(const double* pinv, int investment_count);
void show_investment_total(const double* pinv, int investment_count);
void deallocate_array(double* pinv);

int main()
{
    int maximum_investments;
    get_maximum_investments(maximum_investments);
    double* pinvestments = allocate_array(maximum_investments); // Memory was allocated successfully
    int investment_count;
    get_investments(pinvestments, maximum_investments, investment_count);
    show_investments(pinvestments, investment_count);
    show_investment_total(pinvestments, investment_count);
    deallocate_array(pinvestments);

    return 0;
}

void get_maximum_investments(int& max_investments)
{
    cout << "Enter the maximum number of investments==>";
    cin >> max_investments;
    if (max_investments <= 0)
    {
        cerr << "\nMaximum number of investments must be positive.\n";
        exit(EXIT_FAILURE);
    }
}

double* allocate_array(int max_investments)
{
    double* pinv = new double [max_investments]; // Allocate memory for investments
    if (!pinv) // Did new fail?
    {
        cerr << "Not able to allocate memory for the array\n";
        exit(EXIT_FAILURE);
    }
    return pinv;
}
void get_investments(double* pinv, int max_investments, int& investment_count)
{
    cout << "Enter a maximum of " << max_investments << " investments.";
    cout << "Enter the first investment (q to QUIT): ";
    int i = 0;
    // Using array notation with pointer pinv
    while (i < max_investments && cin >> pinv[i])
    {
        i++;
        if (i < max_investments)
            cout << "Enter the next investment (q to QUIT): ";
    }
    investment_count = i; // Set the actual number of investments
    if (!investment_count)
    {
        cerr << "You entered no investments.";
        delete [] pinv;
        exit(EXIT_FAILURE);
    }
}

void show_investments(const double* pinv, int investment_count)
{
    cout << fixed << setprecision(2)
    << "Your " << investment_count << " investment";
    if (investment_count > 1) "s are" : " is") << ":\n";
    for (int i = 0; i < investment_count; i++)
        cout << setw(21) << pinv[i] << '\n'; // Output the investments
}

void show_investment_total(const double* pinv, int investment_count)
{
    double investment_sum = 0;
    for (int i = 0; i < investment_count; i++)
        investment_sum += pinv[i]; // Accumulate the investments
    cout << "Investment\ntotal: " << setw(14) << investment_sum << '\n';
}

void deallocate_array(double* pinv)
{
    delete [] pinv; // Deallocate memory for the investments
Run 1:
Enter the maximum number of investments==6
Enter a maximum of 6 investments.
Enter the first investment (q to QUIT): 1000.99
Enter the next investment (q to QUIT): 9456.31
Enter the next investment (q to QUIT): 5000
Enter the next investment (q to QUIT): 934.50
Enter the next investment (q to QUIT): q
Your 4 investments are:
  1000.99
  9456.31
  5000.00
  934.50
Investment
  total: 16391.80

Run 2:
Enter the maximum number of investments==3
Enter a maximum of 3 investments.
Enter the first investment (q to QUIT): 20000
Enter the next investment (q to QUIT): 234500.89
Enter the next investment (q to QUIT): 45000.55
Your 3 investments are:
  20000.00
  234500.89
  45000.55
Investment
  total: 299501.44

Run 3:
Enter the maximum number of investments==5
Enter a maximum of 5 investments.
Enter the first investment (q to QUIT): q
You entered no investments.

Run 4:
Enter the maximum number of investments==0
Maximum number of investments must be positive.