

Beginning 1 - string

```
#include <iostream>
#include <cstring>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    char* stringToSearch = new char[40];
    char* quit = new char[10];

    while (strcmp("yes",quit) != 0){
        cout << "\nEnter string: ";
        cin >> stringToSearch;
        cout << "\nChar to find: ";
        cin >> findchar;
        int lenone;
        lenone = (int) strlen(stringToSearch);

        int sum;
        int i;

        sum = 0;

        for(i=0;i<lenone;i++){
            if(stringToSearch[i] == findchar) sum++;
        }

        cout<< "\n " <<<sum;

        cout << "\nQuit?: ";
        cin >> quit;
    }
    return 0;
}
```

2 Beginning — Trajectory

The purpose of this program is to compute the trajectory of a thrown ball. You are given as input three floating point values: an initial height in meters, an initial horizontal velocity in meters per second, and an initial vertical velocity in meters per second. You may assume that both the initial height and the initial horizontal velocity are positive; however, any value is allowable for the initial vertical velocity (a negative value indicates downward motion). The initial position of the ball is $(0, h)$, where h is the initial height. The output is the position of the ball after each second of elapsed time, until the vertical position becomes zero or negative; this final position is to be included in the output. Given a position (x, y) and current horizontal and vertical velocities, the position and velocity after one second can be computed as follows. The new x -coordinate is obtained by adding the current horizontal velocity to x , and the new y -coordinate is obtained by adding the current vertical velocity to $y - 5$. The vertical velocity is then decreased by 10, whereas the horizontal velocity remains unchanged.

Example 1:

```
Enter initial height: 1.8
Enter horizontal velocity: 15.8
Enter vertical velocity: 11.2
```

```
(15.8, 8.0)
(31.6, 4.2)
(47.4, -9.6)
```

Example 2:

```
Enter initial height: 22.2
Enter horizontal velocity: 3.5
Enter vertical velocity: -2.1
```

```
(3.5, 15.1)
(7.0, -2.0)
```

2 Beginning — Trajectory Test Cases

Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter initial height: 1.9
Enter horizontal velocity: 5.5
Enter vertical velocity: 22.7

(5.5, 19.6)
(11.0, 27.300001)
(16.5, 25.000002)
(22.0, 12.700003)
(27.5, -9.599997)

Test Case 2:

Enter initial height: 99.9
Enter horizontal velocity: 1
Enter vertical velocity: 0

(1.0, 94.9)
(2.0, 79.9)
(3.0, 54.9)
(4.0, 19.900002)
(5.0, -25.099998)

Retest: Do the above tests, plus:

Test Case 3:

Enter initial height: 0.1
Enter horizontal velocity: 2.3
Enter vertical velocity: 26.8

(2.3, 21.9)
(4.6, 33.699997)
(6.8999996, 35.499996)
(9.2, 27.299995)
(11.5, 9.099995)
(13.8, -19.100006)

Nov 04, 07 20:17

TrajectoryB.java

Page 1/1

```
// Beginning 2 - Trajectory

import java.io.*;

public class TrajectoryB {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter initial height: ");
        float y = Float.parseFloat(in.readLine());
        System.out.print("Enter horizontal velocity: ");
        float hvel = Float.parseFloat(in.readLine());
        System.out.print("Enter vertical velocity: ");
        float vvel = Float.parseFloat(in.readLine());
        System.out.println();
        float x = 0;
        while (y > 0) {
            x += hvel;
            y += vvel - 5;
            vvel = vvel - 10;
            System.out.println("(" + x + ", " + y + ")");
        }
    }
}
```

Beginning 3 Pyramid

```
#include <iostream>
#include <cstring>
#include <mscorlib.dll>
using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    double pyramidheight;
    double basewidth;
    double trigheight;
    char* quit = new char[20];
    while (strcmp("yes",quit) != 0){
        cout<< "\nEnter height of pyramid: ";
        cin >> pyramidheight;
        cout << "\nEnter width of base: ";
        cin >> basewidth;

        trigheight = Math::Pow(pyramidheight*pyramidheight +
            .25*basewidth*basewidth, 0.5);

        double area;

        area = 4*.5 * trigheight * basewidth + basewidth*basewidth;

        cout<< "\n " <<area;

        cout << "\nQuit?: ";
        cin >> quit;
    }
    return 0;
}
```

4 Beginning — Area Under a Curve

You are to write a program that approximates the area bounded by:

- the curve $y = 1/x$ above;
- the x -axis below;
- the line $x = a$, where $a > 0$ is given as input, on the left; and
- the line $x = b$, where $b > a$ is given as input, on the right (see Figure 1 below).

You are given two floating-point numbers, a and b , and an integer n as input. To approximate the given area, you are to compute the sum of the areas of the n rectangles as shown in Figure 2. Note that all rectangles have the same width, and that the sum of these widths is $b - a$. The height of each rectangle is $1/c$, where the left edge of the rectangle is the line $x = c$.

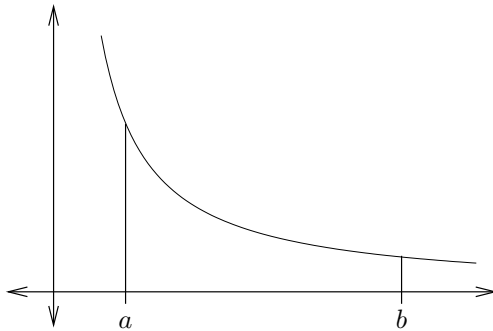


Figure 1

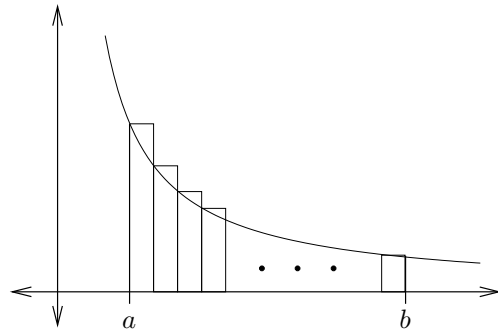


Figure 2

Example:

```
Enter left edge: 0.5
Enter right edge: 1.5
Enter number of rectangles: 2
```

```
The area is: 1.5
```

4 Beginning — Area Under a Curve Test Cases

Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter left edge: 0.001
Enter right edge: 1
Enter number of rectangles: 1

The area is: 998.99994

Test Case 2:

Enter left edge: 1.5
Enter right edge: 10.5
Enter number of rectangles: 10000

The area is: 1.9461659

Retest: Do the above tests, plus the following:

Test Case 3:

Enter left edge: 0.001
Enter right edge: 1
Enter number of rectangles: 10000

The area is: 6.958492

Nov 04, 07 20:17

AreaB.java

Page 1/1

```
// Beginning 4 - Area Under a Curve

import java.io.*;

public class AreaB {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter left edge: ");
        float a = Float.parseFloat(in.readLine());
        System.out.print("Enter right edge: ");
        float b = Float.parseFloat(in.readLine());
        System.out.print("Enter number of rectangles: ");
        int n = Integer.parseInt(in.readLine());
        float w = (b-a)/n;
        float sum = 0;
        for (int i = 0; i < n; i++) {
            sum += w/(a+i*w);
        }
        System.out.println();
        System.out.println("The area is: " + sum);
    }
}
```

Beginning 5 – Infinite Series

Infinite series are used to calculate many things. Most infinite series have terms of the form $a \cdot n \cdot x^{(n-1)}$. For example, $S = 1 + 2 \cdot (.5) + 3 \cdot (.5^2) + 4 \cdot (.5^3) + \dots$. Many infinite series have terms that eventually become increasingly smaller. Sometimes, the sum of the partial series is calculated until the terms become less than a given value. This task will deal with series that are well-behaved.

Accept values for the 'a' value, the 'x' value and the required maximum value of a term. Printout the number of terms before the difference is less than the max value, and the value of the series so far.

Example 1:

Enter the 'x' value for the series: .5

Enter the 'a' value for the series: 2

Enter the max term: .25

Output - Number of terms: 7

Output - The value of the series so far: 7.71875

Example 2:

Enter the 'x' value for the series: .3

Enter the 'a' value for the series: 2

Enter the max term: .005

Output - Number of terms: 8

Output - The value of the series so far: 4.081

Example 3:

Enter the 'x' value for the series: .99

Enter the 'a' value for the series: 1

Enter the max term: .001

Output - Number of terms: 1410

Output - The value of the series so far: 9999.89

Beginning 5 – series

```
#include <iostream>
#include <cstring>
#using <mscorlib.dll>
using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    double delta;
    double xvalue;
    double avalue;

    double termN;
    double termNplus1;
    double xterm;
    double sum;
    int n;

    char* quit = new char[20];

    while (strcmp("yes",quit) != 0){
        cout<< "\nEnter X value: ";
        cin >> xvalue;
        cout << "\nEnter a value: ";
        cin >> avalue;
        cout << "\nEnter maximum delta: ";
        cin >> delta;

        termN = avalue;
        termNplus1 = 2*avalue*xvalue;
        sum = termN + termNplus1;
        xterm = xvalue;
        n = 2;
        while (termNplus1 > delta) {
            n++;
            termN = termNplus1;
            xterm = xterm*xvalue;
            termNplus1 = n*avalue*xterm;
            sum = sum + termNplus1;
        }
        cout<<"\n sum is "<< sum;

        cout << "\n n is " << n;
        cout << "\n diff is "<< termNplus1;

        cout << "\nQuit?: ";
        cin >> quit;
    }
    return 0;
}
```

6 Beginning — Interest

You are given as input a principal, an annual interest rate (as a percentage), and a monthly payment, all of which are floating-point numbers, plus an integer number of months. You are to print the remaining principal after each month, up to the given number of months, assuming that the given monthly payment was made.

From each month's payment, a certain amount will go toward paying interest, and the rest will go toward principal. To compute the amount of interest to be paid, it will first be necessary to convert the annual interest rate from a percentage to a fraction representing the monthly rate. For example, 7.2% annual interest would be 0.6% monthly interest, which is 0.006 when expressed as a fractional value. Once the interest is so converted, this value must be multiplied by the remaining principal in order to obtain the interest payment. Thus, continuing the above example, if the remaining principal is \$10000.00, the interest payment would be \$60.00.

You may assume that all input values are positive, and that the monthly payment is more than the first interest payment. You may also assume that the principal will not be completely paid off before the given number of months is complete. Your output should be printed as floating-point values. Do not truncate any digits to the right of the decimal point.

Example:

```
Enter principal: 10000.00
Enter annual interest rate: 7.2
Enter monthly payment: 500.00
Enter number of months: 5

Principal after 1 months: 9560.0
Principal after 2 months: 9117.36
Principal after 3 months: 8672.064
Principal after 4 months: 8224.098
Principal after 5 months: 7773.4424
```

6 Beginning — Interest Test Cases

Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter principal: 98.76
Enter annual interest rate: 7.5
Enter monthly payment: 12.5
Enter number of months: 4

Principal after 1 months: 86.87726
Principal after 2 months: 74.92024
Principal after 3 months: 62.888496
Principal after 4 months: 50.78155

Test Case 2:

Enter principal: 250000
Enter annual interest rate: 5.5
Enter monthly payment: 1150
Enter number of months: 5

Principal after 1 months: 249995.84
Principal after 2 months: 249991.67
Principal after 3 months: 249987.47
Principal after 4 months: 249983.25
Principal after 5 months: 249979.02

Retest: Do the above tests, plus the following.

Test Case 3:

Enter principal: 1234.56
Enter annual interest rate: 19.9
Enter monthly payment: 25
Enter number of months: 4

Principal after 1 months: 1230.0332
Principal after 2 months: 1225.4313
Principal after 3 months: 1220.753
Principal after 4 months: 1215.9972

Nov 05, 07 10:11

Interest.java

Page 1/1

```
// Beginning 6 - Interest

import java.io.*;

public class Interest {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter principal: ");
        float prin = Float.parseFloat(in.readLine());
        System.out.print("Enter annual interest rate: ");
        float interest = Float.parseFloat(in.readLine())/1200;
        System.out.print("Enter monthly payment: ");
        float pymt = Float.parseFloat(in.readLine());
        System.out.print("Enter number of months: ");
        int n = Integer.parseInt(in.readLine());
        System.out.println();
        for (int i = 1; i <= n; i++) {
            prin = (1 + interest)*prin - pymt;
            System.out.println("Principal after " + i +
                " months: " + prin);
        }
    }
}
```

Advanced 1 String

```
#include <iostream>
#include <cstring>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    char* stringone = new char[40];
    char* stringtwo = new char[40];

    char* quit = new char[10];

    while (strcmp("yes",quit) != 0){
        cout << "\nstring one: ";
        cin >> stringone;
        cout << "\nstring two: ";
        cin >> stringtwo;
        int lenone;
        lenone = (int) strlen(stringone);
        int lentwo;
        lentwo = (int) strlen(stringtwo);
        cout << "\n " << lenone << " " << lentwo;

        int sum;
        int i, j;
        int match;

        sum = 0;

        for(i=0;i<lenone;i++){
            if(stringone[i] == stringtwo[0]) {
                match=1;
                for(j=1;j<lentwo;j++){
                    if(stringone[i+j] != stringtwo[j]){match = 0;}
                }
                if(match == 1)sum++;
                match=0;
            }
        }

        cout<< "\n " << sum;

        cout << "\n" << stringone;
        cout << "\n" << stringtwo;

        cout << "\nQuit?: ";
        cin >> quit;
    }
    return 0;
}
```


2 Advanced — Trajectory

The purpose of this program is to simulate the trajectory of a bouncing ball. You are given as input three floating point numbers: an initial height in meters, an initial velocity in meters per second, and an elasticity factor, plus an integer number of seconds for the simulation. You may assume that the height, elasticity, and number of seconds are all positive, and that the elasticity is strictly less than 1. However, the initial velocity may be any value (a negative value indicates downward motion). Your simulation is to compute a new height and velocity every 0.01 seconds for the given number of seconds.

For a given current height and velocity, the height and velocity after 0.01 seconds are to be computed as follows. The new height is obtained by adding the current velocity multiplied by 0.01. If this new height is zero or negative, you must simulate a bounce. This is done by setting the height to zero, and changing the velocity by reversing its direction and multiplying by the elasticity. Finally, in order to account for gravity, the new velocity is obtained by subtracting 0.1.

Your output must consist of the height of the ball after each second of elapsed time, followed by a report of the total number of bounces.

Example:

```
Enter initial height: 1.15
Enter initial velocity: -20.0
Enter elasticity: .75
Enter number of seconds: 5
```

```
9.987486
10.912465
1.8374404
6.171584
3.9153612
Number of bounces: 2
```

2 Advanced — Trajectory Test Cases

Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter initial height: 1.15
Enter initial velocity: -20
Enter elasticity: .75
Enter number of seconds: 7

9.987486
10.912465
1.8374404
6.171584
3.9153612
3.2507114
0.9553899
Number of bounces: 3

Test Case 2:

Enter initial height: 10
Enter initial velocity: 0
Enter elasticity: .9
Enter number of seconds: 10

5.050001
5.6492033
7.4892063
0.55245036
6.3514504
2.150453
4.8394623
2.0153663
3.9783504
0.29504332
Number of bounces: 4

Retest: Run the above tests, plus:

Test Case 3:

Enter initial height: 1.95
Enter initial velocity: 20
Enter elasticity: .8
Enter number of seconds: 7

16.99998
22.04994
17.099903
2.149842
10.947045
13.797084
6.6471243
Number of bounces: 1

Nov 04, 07 20:26

TrajectoryA.java

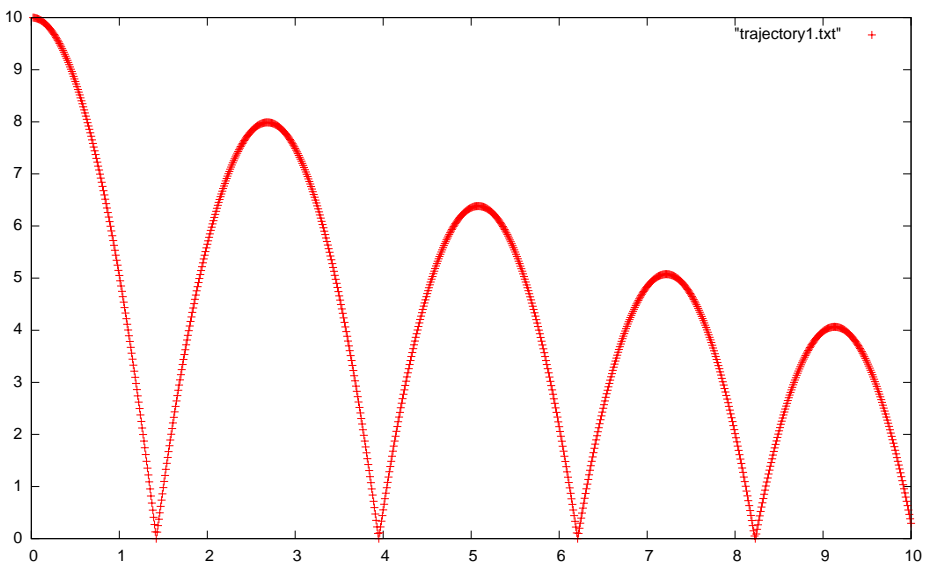
Page 1/1

```
// Advanced 2 - Trajectory

/*
 * This code contains several comments containing print statements. If
 * these lines are uncommented, the program will produce plot data that
 * can be processed by gnuplot (http://www.gnuplot.info/).
 */

import java.io.*;
public class TrajectoryA {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
//        System.out.print("# ");
        System.out.print("Enter initial height: ");
        float h = Float.parseFloat(in.readLine());
        System.out.print("Enter initial velocity: ");
        float v = Float.parseFloat(in.readLine());
        System.out.print("Enter elasticity: ");
        float e = Float.parseFloat(in.readLine());
        System.out.print("Enter number of seconds: ");
        int n = Integer.parseInt(in.readLine());
        System.out.println();
        int bounces = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < 100; j++) {
                h += v*0.01;
                if (h <= 0) {
                    h = 0;
                    v = -v*e;
                    bounces++;
                }
                v = v - 0.1f;
//                System.out.println((i+j*0.01+0.01) + " " + h);
            }
//            System.out.print("# ");
            System.out.println(h);
        }
//        System.out.print("# ");
        System.out.println("Number of bounces: " + bounces);
    }
}
```



4 Advanced — Area Under a Curve

You are to write a program that approximates the area bounded by:

- the curve $y = 1/x$ above;
- the x -axis below;
- the line $x = a$, where $a > 0$ is given as input, on the left; and
- the line $x = b$, where $b > a$ is given as input, on the right (see Figure 1 below).

You are given three floating-point numbers, a , b , and ϵ , as input. You may assume that all three numbers are positive. To approximate the given area, you are to compute the sum of the areas of a series of rectangles as shown in Figure 2. Note that all rectangles have the same width, and that the sum of these widths is $b - a$. The height of each rectangle is $1/c$, where the left edge of the rectangle is the line $x = c$. As a first approximation, use a single rectangle with width $a - b$ and height $1/a$. Then find better approximations by doubling the number of rectangles, finishing when you find two successive approximations that differ by no more than ϵ . Print out only the final approximation and the number of rectangles used to obtain this approximation.

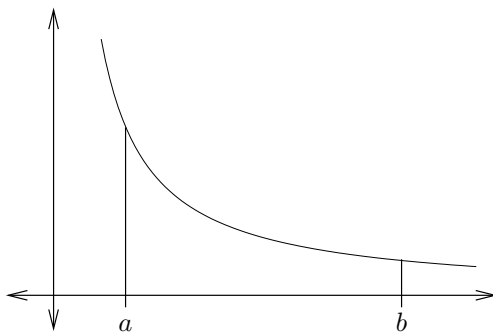


Figure 1

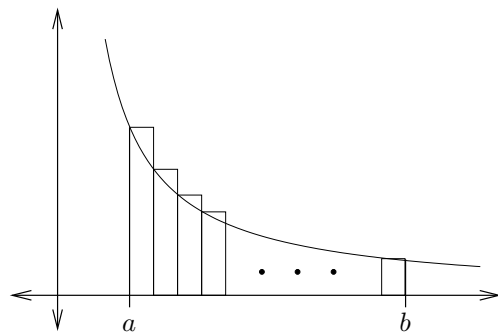


Figure 2

Example:

```
Enter left edge: 0.5
Enter right edge: 1.5
Enter epsilon: .01

Area: 1.1038384
Number of rectangles: 128
```

4 Advanced — Area Under a Curve Test Cases

Note: Area should be accurate to 3 significant digits.

Test Case 1:

Enter left edge: 111.1
Enter right edge: 155.5
Enter epsilon: .1

Area: 0.36636162
Number of rectangles: 2

Test Case 2:

Enter left edge: .1
Enter right edge: 2
Enter epsilon: .001

Area: 2.9962814
Number of rectangles: 16384

Retest: Do the above test, plus the following.

Test Case 3:

Enter left edge: .1
Enter right edge: 10
Enter epsilon: .01

Area: 4.61116
Number of rectangles: 8192

Nov 05, 07 8:06

AreaA.java

Page 1/1

```
// Advanced 4 - Area Under a Curve

import java.io.*;

public class AreaA {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter left edge: ");
        float a = Float.parseFloat(in.readLine());
        System.out.print("Enter right edge: ");
        float b = Float.parseFloat(in.readLine());
        System.out.print("Enter epsilon: ");
        float epsilon = Float.parseFloat(in.readLine());
        int n = 1;
        float sum = (b-a)/a;
        float last;
        do {
            n = n*2;
            last = sum;
            sum = 0;
            float w = (b-a)/n;
            for (int i = 0; i < n; i++) {
                sum += w/(a+i*w);
            }
        } while (Math.abs(sum - last) > epsilon);
        System.out.println();
        System.out.println("Area: " + sum);
        System.out.println("Number of rectangles: " + n);
    }
}
```

Advanced 5 – Infinite Series

Infinite series are used to calculate many things, including the expected number of rolls before a specific roll of the dice occurs. Most infinite series have terms of the form $a \cdot n \cdot x^{(n-1)}$. For example, $S = 1 + 2 \cdot (.5) + 3 \cdot (.5^2) + 4 \cdot (.5^3) + \dots$

For calculating the expected number of rolls, the 'n' will be the number of rolls, and the 'a' will be the probability of the desired (or undesired) roll to occur, and the x will be equal to (1-a) and will be the probability of that specified roll of the dice not happening on the previous n-1 rolls. Consider the expected time to roll a "6" with one die. The probability of it happening on the 1st roll is 1/6 so the first term is $1 \cdot 0.1667$. The prob for it happening on the second roll is $1/6 \cdot (1-1/6)$ or $.1667 \cdot .8333$. The contribution to the expected number of rolls is $2 \cdot .1667 \cdot .8333$.

This infinite series has terms that are eventually decreasing. In this problem, this series will be calculated until the **terms are decreasing** and the value of the new term is less than a given value.

The problem is calculating the expected number of rolls of two dice before a specified number sum (e.g. 11) appears on the faces of the dice.

Accept the specified value and the required maximum value of the new term. Printout the number of terms before the terms are decreasing and the value is less than the max value, the term value, and the estimate so far of the expected number of rolls.

Example 1:

Enter the specified value for the series: 11

Enter the max term: .25

Number of terms: 39

The last term: .2468

The value of the series so far: 11.865

Example 2:

Enter the specified value for the series: 8

Enter the max term: .05

Number of terms: 31

The last term: .048

The value of the series so far: 6.829

5Advanced - Series estimation

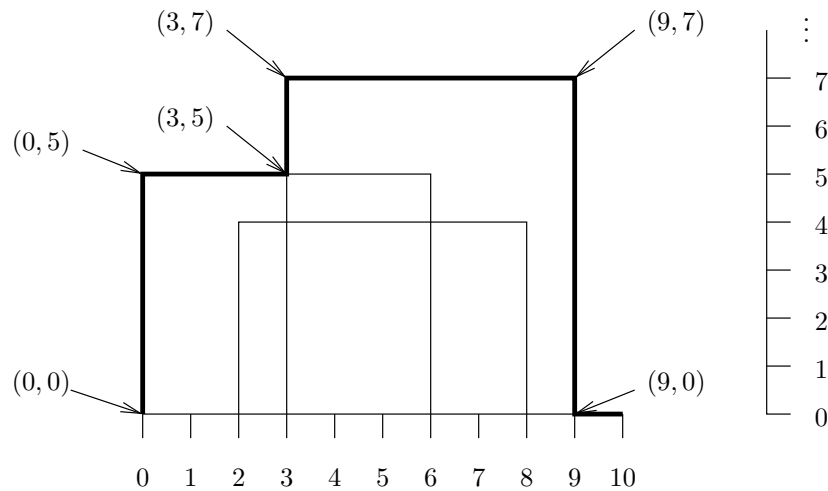
```
#include <iostream>
#include <cstring>
#include <mscorlib.dll>
using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    double delta, dicevalue, xvalue, avalue;
    double termN;
    double termNplus1;
    double xterm;
    double sum;
    int n;
    char* quit = new char[20];
    while (strcmp("yes",quit) != 0){
        cout<< "\nEnter two-dice value: ";
        cin >> dicevalue;
        if(dicevalue < 2 || dicevalue > 12) {
            cout<< "\n Illegal value for two dice";
            break;
        }
        if(dicevalue < 8) {
            avalue = (dicevalue-1)/36;
        }
        else {
            avalue = (13-dicevalue)/36;
        }

        xvalue = 1 - avalue;
        cout << "\nEnter maximum delta: ";
        cin >> delta;
        termN = avalue;
        termNplus1 = 2*avalue*xvalue;
        sum = termN + termNplus1;
        xterm = xvalue;
        n = 2;
        while (termNplus1 > termN || termNplus1 > delta) {
            n++;
            termN = termNplus1;
            xterm = xterm*xvalue;
            termNplus1 = n*avalue*xterm;
            sum = sum + termNplus1;
            cout<<"\nsum is "<<sum << "  n is  " << n;
        }
        cout<<"\n average number of tries is "<< sum;
        cout << "\n n is " << n;
        cout << "\n term is "<< termNplus1;
        cout << "\nQuit?: ";
        cin >> quit;
    }
    return 0;
}
```


6 Advanced — Manhattan Skyline

You are given as input an integer number of buildings, followed by a description of the shape of each of these buildings. Each building is a rectangle described by the position of its left edge, the position of its right edge, and its height, all of which are integers. The positions of the edges are all at least 0 and no more than 10, whereas the heights may be any positive integer. You are to print a description of the skyline formed by these building (see figure below). Specifically, you are to print the sequence of corners of the skyline from left to right. The first corner should be at position $(l,0)$, where l is the leftmost left edge, and the last corner should be $(r,0)$, where r is the rightmost right edge.



Example:

```
Enter number of buildings: 3
Enter left edge 0: 0
Enter right edge 0: 6
Enter height 0: 5
Enter left edge 1: 2
Enter right edge 1: 8
Enter height 1: 4
Enter left edge 2: 3
Enter right edge 2: 9
Enter height 2: 7
```

```
(0, 0)
(0, 5)
(3, 5)
(3, 7)
(9, 7)
(9, 0)
```

6 Advanced — Manhattan Skyline Test Cases

Test Case 1:

Enter number of buildings: 3
Enter left edge 0: 3
Enter right edge 0: 6
Enter height 0: 8
Enter left edge 1: 0
Enter right edge 1: 4
Enter height 1: 5
Enter left edge 2: 5
Enter right edge 2: 8
Enter height 2: 10

(0, 0)
(0, 5)
(3, 5)
(3, 8)
(5, 8)
(5, 10)
(8, 10)
(8, 0)

Test Case 2:

Enter number of buildings: 4
Enter left edge 0: 6
Enter right edge 0: 7
Enter height 0: 11
Enter left edge 1: 2
Enter right edge 1: 3
Enter height 1: 12
Enter left edge 2: 1
Enter right edge 2: 10
Enter height 2: 7
Enter left edge 3: 4
Enter right edge 3: 8
Enter height 3: 3

(1, 0)
(1, 7)
(2, 7)
(2, 12)
(3, 12)
(3, 7)
(6, 7)
(6, 11)
(7, 11)
(7, 7)
(10, 7)
(10, 0)

Retest: Do the above tests, plus the following.

Test Case 3:

Enter number of buildings: 2
Enter left edge 0: 4
Enter right edge 0: 10
Enter height 0: 25
Enter left edge 1: 0
Enter right edge 1: 6
Enter height 1: 21

(0, 0)
(0, 21)
(4, 21)
(4, 25)
(10, 25)
(10, 0)

Nov 05, 07 11:21

ManhattanSkyline.java

Page 1/1

```
// Advanced 6 - Manhattan Skyline

import java.io.*;

public class ManhattanSkyline {

    public static void main(String[] args) throws Exception {
        BufferedReader in
            = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter number of buildings: ");
        int n = Integer.parseInt(in.readLine());
        int h[] = new int[10];

        for (int i = 0; i < 10; i++) {
            h[i] = 0;
        }
        for (int i = 0; i < n; i++) {
            System.out.print("Enter left edge " + i + ":");
            int left = Integer.parseInt(in.readLine());
            System.out.print("Enter right edge " + i + ":");
            int right = Integer.parseInt(in.readLine());
            System.out.print("Enter height " + i + ":");
            int height = Integer.parseInt(in.readLine());
            for (int j = left; j < right; j++) {
                if (h[j] < height) h[j] = height;
            }
        }

        System.out.println();
        if (h[0] > 0) {
            System.out.println("(0,0)");
            System.out.println("(0, " + h[0] + ")");
        }
        for (int i = 1; i < 10; i++) {
            if (h[i] != h[i-1]) {
                System.out.println("(" + i + ", " + h[i-1] + ")");
                System.out.println("(" + i + ", " + h[i] + ")");
            }
        }
        if (h[9] > 0) {
            System.out.println("(10, " + h[9] + ")");
            System.out.println("(10,0)");
        }
    }
}

```