

B1 Geometric Objects – Problem Statement

This program will read in descriptions of three geometric objects (squares, rectangles, and triangles). The program will calculate the area of each object and determine which object has the greatest area. It will return the number of the largest object (1, 2 or 3), the type ('s', 'r', or 't') and the area.

Formulas;

Area of square is side*side

Area of rectangle is base*height

Area of triangle is $\frac{1}{2}$ * base * height

Prompts can be on separate lines.

Example:

Enter type of first object: t enter height of triangle: 10 enter base of triangle: 10

Enter type of second object: s enter length of square: 10.5

Enter type of third object: r enter height of rectangle: 10 enter width of rectangle: 5

The largest is object 2 of type s with area 110.25

B1 Geometric Objects – Test Cases

Prompts can be on separate lines. Do not be concerned with exact wording of prompts

First round testing

Test 1:

Enter type of first object: t enter height of triangle: 21 enter base of triangle: 11
Enter type of second object: s enter length of square: 10
Enter type of third object: r enter height of rectangle: 10 enter width of rectangle: 5
The largest is object 1 of type t with area 115.5

Test 2:

Enter type of first object: r enter height of rectangle: 9 enter width of rectangle: 9
Enter type of second object: t enter height of triangle: 20 enter base of triangle: 8
Enter type of third object: s enter length of square: 10
The largest is object 3 of type s with area 100

Test 3:

Enter type of first object: t enter height of triangle: 10 enter base of triangle: 10
Enter type of second object: r enter height of rectangle: 10 enter width of rectangle: 10
Enter type of third object: r enter height of rectangle: 10 enter width of rectangle: 9
The largest is object 2 of type r with area 100

Second round testing – do all the above plus the following tests

Test 4:

Enter type of first object: t enter height of triangle: 10 enter base of triangle: 10
Enter type of second object: s enter length of square: 10
Enter type of third object: r enter height of rectangle: 10 enter width of rectangle: 5
The largest is object 2 of type s with area 100

Test 5:

Enter type of first object: t enter height of triangle: 20 enter base of triangle: 10.75
Enter type of second object: s enter length of square: 10
Enter type of third object: r enter height of rectangle: 10 enter width of rectangle: 5
The largest is object 1 of type t with area 107.5

B1GeoArea Code

```
// GeoAreaB.cpp : The program accepts input about three geometric
//             objects (squares, rectangles, and triangles). The program
//             calculates the area of each object and return the number, type,
//             and area of the largest.

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    char type1, type2, type3;
    float length1, length2;
    float area1, area2, area3;
    cout<< "\nEnter type 1: ";
    cin>> type1;
    if(type1== 's' || type1 == 'S')
    {
        cout<< "\nEnter one length:";
        cin>> length1;
        area1 = length1 * length1;
    }
    if(type1== 'r' || type1 == 'R')
    {
        cout<< "\nEnter height:"; cin>> length1;
        cout<< "\nEnter length:"; cin>> length2;
        area1 = length1 * length2;
    }
    if(type1== 't' || type1 == 'T')
    {
        cout<< "\nEnter height of triangle:"; cin>> length1;
        cout<< "\nEnter length of base:"; cin>> length2;
        area1 = .5 * length1 * length2;
    }

    cout<< "\nEnter type 2: ";
    cin>> type2;
    if(type2== 's' || type2 == 'S')
    {
        cout<< "\nEnter one length:";
        cin>> length1;
        area2 = length1 * length1;
    }
    if(type2== 'r' || type2 == 'R')
    {
        cout<< "\nEnter height:"; cin>> length1;
        cout<< "\nEnter length:"; cin>> length2;
        area2 = length1 * length2;
    }
    if(type2== 't' || type2 == 'T')
    {
        cout<< "\nEnter height of triange:"; cin>> length1;
```

```

        cout<< "\nEnter length of base:";  cin>> length2;
        area3 = .5 * length1 * length2;
    }
    cout<< "\nEnter type 3: ";
    cin>> type3;
    if(type3== 's' || type3 == 'S')
    {
        cout<< "\nEnter one length:";
        cin>> length1;
        area3 = length1 * length1;
    }
    if(type3== 'r' || type3 == 'R')
    {
        cout<< "\nEnter height:"; cin>> length1;
        cout<< "\nEnter length:"; cin>> length2;
        area3 = length1 * length2;
    }
    if(type3== 't' || type3 == 'T')
    {
        cout<< "\nEnter height of triangle:"; cin>> length1;
        cout<< "\nEnter length of base:"; cin>> length2;
        area3 = .5 * length1 * length2;
    }

    if (area1 >= area2 && area1 >= area3)
    {
        cout<< "\nThe largest figure is number 1 of type "
        << type1 << " with an area of " << area1;
    }
    if (area2 >= area1 && area2 >= area3)
    {
        cout<< "\nThe largest figure is number 2 of type " << type2
        << " with an area of " << area2;
    }
    if (area3 >= area1 && area3 >= area2)
    {
        cout<< "\nThe largest figure is number 3 of type "
        << type3 << " with an area of " << area3;
    }
    return 0;
}

```

2 Beginning — Optimal Chocolate Purchase

We have a certain amount of money that we wish to use to buy chocolate bars. There are two sizes of chocolate bars, and we wish to buy a combination of these two sizes that maximizes the amount of chocolate without exceeding the amount of money that we have. Write a program that takes an amount of money in cents (so that you only need integer values), the cost in cents of the two sizes, and the two sizes in ounces. The program must then display the number of each size, the total weight, and the total cost for an optimal purchase. You may assume that all inputs are positive integers.

Example 1:

```
Enter the amount of money (in cents): 575
Enter the cost of the first candy bar (in cents): 50
Enter the size of the first candy bar (in ounces): 3
Enter the cost of the second candy bar (in cents): 100
Enter the size of the second candy bar (in ounces): 7
```

```
Buy 1 of the first candy bar and 5 of the second.
Total size: 38 ounces
Total cost: 550 cents
```

Example 2:

```
Enter the amount of money (in cents): 1100
Enter the cost of the first candy bar (in cents): 109
Enter the size of the first candy bar (in ounces): 17
Enter the cost of the second candy bar (in cents): 79
Enter the size of the second candy bar (in ounces): 13
```

```
Buy 2 of the first candy bar and 11 of the second.
Total size: 177 ounces
Total cost: 1087 cents
```

2 Beginning — Optimal Chocolate Purchase Test Cases

Test Case 1:

Enter the amount of money (in cents): 1100
Enter the cost of the first candy bar (in cents): 109
Enter the size of the first candy bar (in ounces): 17
Enter the cost of the second candy bar (in cents): 79
Enter the size of the second candy bar (in ounces): 13

Buy 2 of the first candy bar and 11 of the second.
Total size: 177 ounces
Total cost: 1087 cents

Test Case 2:

Enter the amount of money (in cents): 988
Enter the cost of the first candy bar (in cents): 46
Enter the size of the first candy bar (in ounces): 7
Enter the cost of the second candy bar (in cents): 85
Enter the size of the second candy bar (in ounces): 13

Buy 3 of the first candy bar and 10 of the second.
Total size: 151 ounces
Total cost: 988 cents

Second Submission: Do the above tests, plus the following:

Test Case 3:

Enter the amount of money (in cents): 540
Enter the cost of the first candy bar (in cents): 119
Enter the size of the first candy bar (in ounces): 11
Enter the cost of the second candy bar (in cents): 69
Enter the size of the second candy bar (in ounces): 5

Buy 4 of the first candy bar and 0 of the second.
Total size: 44 ounces
Total cost: 476 cents

```

/* Beginning 2 - Optimal Chocolate Purchase
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.ChocolateBeginning
{
    class Chocolate
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the amount of money (in cents): ");
            int money = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the first candy bar (in cents): ");
            int cost1 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the size of the first candy bar (in ounces): ");
            int size1 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the second candy bar (in cents): ");
            int cost2 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the size of the second candy bar (in ounces): ");
            int size2 = Int32.Parse(Console.ReadLine());
            Console.WriteLine();
            int num1 = 0;
            int num2 = money / cost2;
            int totalSize = num2 * size2;
            for (int i = 1; i <= money / cost1; i++)
            {
                int j = (money - i * cost1) / cost2;
                if (i * size1 + j * size2 > totalSize)
                {
                    num1 = i;
                    num2 = j;
                    totalSize = num1 * size1 + num2 * size2;
                }
            }
            Console.WriteLine("Buy " + num1 + " of the first candy bar and " +
                num2 + " of the second.");
            Console.WriteLine("Total size: " + totalSize + " ounces");
            Console.WriteLine("Total cost: " + (num1 * cost1 + num2 * cost2) + " cents");
            Console.ReadLine();
        }
    }
}

```

B3TreePath Problem Statement

This program accepts a description of a binary tree. Each node in the tree will be identified by an integer. A triple of numbers represents a node and subnodes in the tree. The first number represents a node in the tree and the following two numbers represent the left subnode and the right subnode.

For example:

1 2 3

2 4 5 represents the tree at the right

3 6 7

The program will accept three triples (as above). Then the program prompts for a path that is specified as a sequence of three nodes. For example, 1 2 4 would be the path that is indicated by the blue arrow above. For a correct path, the program must state that it is correct. Sequence 1 2 7 is not a path in the tree and the program should state that the path is incorrect and third node is not found (do not output the identification number of the node not found).

A binary tree is a special type of graph. Each node has 0 to 2 subnodes below it. There are no loops in a tree and no shared lower trees. Thus there is only one possible path from a higher node to a lower node. In this problem we will only allow downward motion on the tree. A path can start anywhere in the tree but must go downward.

Example 1:

Enter node: 1 enter left subnode: 2 enter right subnode: 3

Enter node: 2 enter left subnode: 4 enter right subnode: 5

Enter node: 4 enter left subnode: 6 enter right subnode: 7

Enter first path node: 2 enter next node: 4 enter next node: 7 Successful path

Example 2:

Enter node: 1 enter left subnode: 2 enter right subnode: 3

Enter node: 2 enter left subnode: 4 enter right subnode: 5

Enter node: 4 enter left subnode: 6 enter right subnode: 7

Enter first path node: 2 enter next node: 3 enter next node: 7 node 2 not found. Incorrect path

Note that "node 2 not found" refers to the second node entered, not the node numbered "2"

B3TreePath Test Cases

The formatting is not important. If the path is not found, the program must indicate which (first, second, third) node is not found on the path.

First round testing

test 1:

Enter node: 8 enter left subnode: 4 enter right subnode: 7
Enter node: 4 enter left subnode: 3 enter right subnode: 5
Enter node: 3 enter left subnode: 6 enter right subnode: 9
Enter first path node: 8 enter next node: 4 enter next node: 5 Successful path

Test 2:

Same input but different path

Enter first path node: 2 enter next node: 3 enter next node: 5 first node not found, incorrect path

Test 3:

Enter node: 1 enter left subnode: 2 enter right subnode: 3
Enter node: 2 enter left subnode: 4 enter right subnode: 5
Enter node: 4 enter left subnode: 6 enter right subnode: 7
Enter first path node: 2 enter next node: 3 enter next node: 7 node 2 not found. Incorrect path

Second round testing – do all the above plus the following tests

Test 4:

Enter node: 1 enter left subnode: 2 enter right subnode: 3
Enter node: 2 enter left subnode: 4 enter right subnode: 5
Enter node: 4 enter left subnode: 6 enter right subnode: 7
Enter first path node: 1 enter next node: 2 enter next node: 7 node 3 not found. Incorrect path

Test 5:

Enter node: 1 enter left subnode: 2 enter right subnode: 3
Enter node: 2 enter left subnode: 4 enter right subnode: 5
Enter node: 3 enter left subnode: 6 enter right subnode: 7
Enter first path node: 1 enter next node: 3 enter next node: 2 node 3 not found. Incorrect path

B3Path Code

```
// PathB.cpp : This program accepts a definition of a binary tree.
// Each node will be identified by an integer. The specification will be
// a triple of numbers. The first number be a node and the following two
// numbers will be subnodes of the first node
// There will be three inputs that show the subnodes of a tree node.
// Then the program will accept a sequence of 3 numbers that represents a
// possible path from a node in the tree to a node below that node. The
// program will indicate whether the sequence is a valid path.

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    int treenode1[3], treenode2[3], treenode3[3];
    int start, end, distance;
    int loc1, loc2, loc3, loc;

    cout<< "\nEnter first node: "; cin>> treenode1[0];
    cout<< "\nEnter left subtree node: "; cin>> treenode1[1];
    cout<< "\nEnter right subtree node: "; cin>> treenode1[2];

    cout<< "\nEnter next node: "; cin>> treenode2[0];
    cout<< "\nEnter left subtree node: "; cin>> treenode2[1];
    cout<< "\nEnter right subtree node: "; cin>> treenode2[2];

    cout<< "\nEnter next node: "; cin>> treenode3[0];
    cout<< "\nEnter left subtree node: "; cin>> treenode3[1];
    cout<< "\nEnter right subtree node: "; cin>> treenode3[2];

    cout<< "\nEnter start location: "; cin>> loc1;
    if (loc1 == treenode1[0])
    {
        cout<< "\nEnter second location: "; cin>> loc2;
        if (loc2 == treenode1[1] || loc2 == treenode1[2])
        {
            //cout<< "\nEnter third location: "; cin>> loc3;
            if (loc2 == treenode2[0])
            {
                cout<< "\nEnter third location: "; cin>> loc3;
                if (loc3 == treenode2[1] || loc3 == treenode2[2])
                {
                    cout << "\nEPath was successful";
                    cin >> loc;
                    return 0;
                }
            }
            cout<< "\nFThird node was not found."
                << " Path is incorrect.";
            cin>>loc;
            return 0;
        }
        if (loc2 == treenode3[0])
```

```

    {
        cout<< "\nGEnter third location: "; cin>> loc3;
        if (loc3 == treenode3[1] || loc3 == treenode3[2])
        {
            cout << "\nHPath was successful";
            cin >> loc;
            return 0;
        }
        cout<< "\nIThird node was not found."
            << " Path is incorrect.";
        cin>>loc;
        return 0;
    }
    cout<<"\nJSecond node was not found. Path is incorrect.";
    cin >>loc;
    return 0;
}

if (loc1 == treenode2[0])
{
    cout<< "\nKEnter second location: "; cin>> loc2;
    if (loc2 == treenode2[1] || loc2 == treenode2[2])
    {
        //cout<< "\nLEnter third location: "; cin>> loc3;
        if (loc2 == treenode1[0])
        {
            cout<< "\nMEnter third location: "; cin>> loc3;
            if (loc3 == treenode1[1] || loc3 == treenode1[2])
            {
                cout << "\nNPath was successful";
                cin >> loc;
                return 0;
            }
            cout<< "\nOThird node was not found."
                <<" Path is incorrect.";
            cin>>loc;
            return 0;
        }
        if (loc2 == treenode3[0])
        {
            cout<< "\nPEnter third location: "; cin>> loc3;
            if (loc3 == treenode3[1] || loc3 == treenode3[2])
            {
                cout << "\nQPath was successful";
                if (loc2 == treenode1[1] || loc2 == treenode1[2])
            {
                //cout<< "\nREnter third location: "; cin>> loc3;
                if (loc2 == treenode2[0])
                {
                    cout<< "\nSEnter third location: "; cin>> loc3;
                    if (loc3 == treenode2[1] || loc3 == treenode2[2])
                    {
                        cout << "\nTPath was successful";
                        cin >> loc;
                        return 0;
                    }
                }
            }
        }
    }
}

```

```

        cout<< "\nUThird node was not found."
            << " Path is incorrect.";
        cin>>loc;
        return 0;
    }
    if (loc2 == treenode3[0])
    {
        cout<< "\nVEnter third location: "; cin>> loc3;
        if (loc3 == treenode3[1] || loc3 == treenode3[2])
        {
            cout << "\nWPath was successful";
            cin >> loc;
            return 0;
        }
        cout<< "\nXThird node was not found."
            <<" Path is incorrect.";
        cin>>loc;
        return 0;
    }
    cout<<"\nYSecond node was not found. Path is incorrect.";
    cin >>loc;
    return 0;
}

        cin >> loc;
        return 0;
    }
    cout<< "\nZThird node was not found. ""
        <<" Path is incorrect.";
    cin>>loc;
    return 0;
}
cout<<"\nAASecond node was not found. Path is incorrect.";
cin >>loc;
return 0;
}

if (loc1 == treenode3[0])
{
    cout<< "\nABEnter second location: "; cin>> loc2;

    if (loc2 == treenode3[1] || loc2 == treenode3[2])
    {
        //cout<< "\nEEnter third location: "; cin>> loc3;
        if (loc2 == treenode1[0])
        {
            cout<< "\nACEnter third location: "; cin>> loc3;
            if (loc3 == treenode1[1] || loc3 == treenode1[2])
            {
                cout << "\nADPath was successful";
                cin >> loc;
                return 0;
            }
            cout<< "\nAETHird node was not found. "
                <<" Path is incorrect.";
            cin>>loc;
        }
    }
}

```

```

        return 0;
    }
    if (loc2 == treenode2[0])
    {
        cout<< "\nAFEnter third location: "; cin>> loc3;
        if (loc3 == treenode2[1] || loc3 == treenode2[2])
        {
            cout << "\nAGPath was successful";
            cin >> loc;
            return 0;
        }
        cout<< "\nAHThird node was not found."
            <<" Path is incorrect.";
        cin>>loc;
        return 0;
    }
    cout<<"\nAISsecond node was not found. Path is incorrect.";
    cin >>loc;
    return 0;
}
}
cout<< "\nAJSequence not valid. first node not found";
cin>>loc1;
return 0;
}

```

4 Beginning — Approximating π

The mathematical constant π can be defined by the following infinite series:

$$\pi = 4 \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \dots$$

Write a program that takes as input a nonnegative integer n and produces as output the approximation of π given by the sum of the first n terms of the above series; for example, for $n = 4$, the program should compute

$$\frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7}.$$

Example 1:

```
Enter n: 1
Pi is approximately 4
```

Example 2:

```
Enter n: 2
Pi is approximately 2.66666666666667
```

Example 3:

```
Enter n: 0
Pi is approximately 0
```

Example 4:

```
Enter n: 1000
Pi is approximately 3.14059265383979
```

Note: Your answers may not be identical to those above, but should agree when rounded to 4 digits (i.e., 3 digits to the right of the decimal point).

4 Beginning — Approximating π Test Cases

The answers produced must match when rounded to 4 digits (i.e., 3 digits to the right of the decimal point).

Test Case 1:

Enter n: 0
Pi is approximately 0

Test Case 2:

Enter n: 10
Pi is approximately 3.0418396189294

Test Case 3:

Enter n: 12345
Pi is approximately 3.14167365804489

Second Submission: Do the above tests plus the following:

Test Case 4:

Enter n: 123
Pi is approximately 3.14972260055572

```
/* Beginning 4 - Approximating Pi
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.PiBeginning
{
    class Pi
    {
        static void Main(string[] args)
        {
            Console.Write("Enter n: ");
            int n = Int32.Parse(Console.ReadLine());
            double numerator = 4.0;
            double pi = 0.0;
            for (int i = 0; i < n; i++)
            {
                pi += numerator / (2 * i + 1);
                numerator *= -1;
            }
            Console.WriteLine("Pi is approximately " + pi);
            Console.ReadLine();
        }
    }
}
```

B5 Tiling Problem Statement

This program must calculate how many tiles are needed to tile a floor. The tiles are 8 inches by 8 inches. Tiles can be used whole or a part of a tile can be used. Only one usable piece can be cut from a tile. That is, if a piece is cut from a tile, the rest of the tile must be thrown away. The program accepts the length and width of the room and returns how many whole tiles are used and how many part tiles are used. The lengths are all in inches.

Example 1:

Enter width of the room in inches: 160

Enter length of the room in inches: 240

The number of whole tiles is 600 part tiles is 0

Example 2:

Enter width of the room in inches: 100

Enter length of the room in inches: 120

The number of whole tiles is 180 part tiles is 15

B5 Tiling Test Cases

The exact format of the prompts is unimportant

First Round

Test 1:

Enter width of the room in inches: 160

Enter length of the room in inches: 240

The number of whole tiles is 600 part tiles is 0

Test 2:

Enter width of the room in inches: 100

Enter length of the room in inches: 100

The number of whole tiles is 144 part tiles is 25

Test 3:

Enter width of the room in inches: 7

Enter length of the room in inches: 8

The number of whole tiles is 0 part tiles is 1

Second Round – repeat tests above and the additional tests below

Test 4:

Enter width of the room in inches: 15

Enter length of the room in inches: 100

The number of whole tiles is 12 part tiles is 14

Test 5:

Enter width of the room in inches: 8

Enter length of the room in inches: 8

The number of whole tiles is 1 part tiles is 0

B5tiling Code

```
// TilingB.cpp : This problem is to calculate how many tiles are needed to
// tile a floor. The tiles are 8 by 8. Tiles can be used whole or a
// part of a tile can be used. Only one usable piece can be cut from a
// tile. That is, if a piece is cut from a tile, the rest of the tile
// must be thrown away. The problem accepts the length and width of the
// room and returns how many whole tiles are used and how many part
// tiles are used.

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    float width, length;
    int wholetiles, parttiles, tilewidth, tilelength;
    int extrawide, extralength;

    cout<<"\nEnter the width of the floor in inches: "; cin >> width;
    cout<< "\nEnter the length of the floor in inches: "; cin >> length;

    tilewidth = (int) width/8; tilelength = (int) length/8;

    parttiles = 0;

    wholetiles = tilewidth * tilelength;
    extrawide = width - 8*tilewidth;
    extralength = length - 8*tilelength;

    if (extrawide > 0) {parttiles = tilelength;}
    if (extralength > 0) {parttiles = parttiles + tilewidth;}
    if (extrawide > 0 && extralength > 0) {parttiles++;}

    cout<< "\nThe number of wholes tiles is " << wholetiles
         << " and the number of parttiles is " << parttiles;

    cout<< "\n enter width"; cin >> width;

    return 0;
}
```

6 Beginning — Subsequence Matching

Write a program that takes a text string and a target string as input, and outputs whether the target string appears as a subsequence of the text string. The subsequence needs to occur in the same order in both strings, but does not necessarily need to be contiguous in the text string; for example, “bnn” occurs as a subsequence of “banana”, but not of “Annabell”. Both strings must be input exactly as the strings to be used, with nothing artificially separating the characters. You may assume that both strings are nonempty and are of length at most 20.

Example 1:

```
Enter text string: banana
Enter target string: bnn
Subsequence found.
```

Example 2:

```
Enter text string: Mississippi
Enter target string: psi
Subsequence not found.
```

Example 3:

```
Enter text string: structure
Enter target string: data
Subsequence not found.
```

6 Beginning — Subsequence Matching Test Cases

Test Case 1:

Enter text string: banana
Enter target string: bnn
Subsequence found.

Test Case 2:

Enter text string: Mississippi
Enter target string: psi
Subsequence not found.

Test Case 3:

Enter text string: aaaabaaaabaaaabaaaab
Enter target string: abcd
Subsequence not found.

Test Case 4:

Enter text string: sandwich
Enter target string: sh
Subsequence found.

Second Submission: Do the above tests, plus the following:

Test Case 5:

Enter text string: abcde
Enter target string: e
Subsequence found.

```

/* Beginning 6 - Subsequence Matching
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.SubsequenceBeginning
{
    class Subsequence
    {
        static void Main(string[] args)
        {
            Console.Write("Enter text string: ");
            string text = Console.ReadLine();
            Console.Write("Enter target string: ");
            string target = Console.ReadLine();
            int j = 0;
            for (int i = 0; i < text.Length; i++)
            {
                if (target[j] == text[i])
                {
                    j++;
                    if (j == target.Length)
                    {
                        Console.WriteLine("Subsequence found.");
                        Console.ReadLine();
                        return;
                    }
                }
            }
            Console.WriteLine("Subsequence not found.");
            Console.ReadLine();
        }
    }
}

```

1A Geometric Objects Problem Statement

This project reads up to ten descriptions of geometric objects. It then finds the object with the largest surface area. It will print out the ordinal number (e.g. 2 for second), the type of object (e.g. 't' for triangle), and the area of the object. The input for each object will be a char (s for square, r for rectangle, c for circle, t for triangle) and the necessary lengths (real values). Only those four types of objects will be used.

Assume that there is at least one object and that the input is syntactically correct. For example, only one length is given for a circle's radius.

Formulas;

Area of square is side*side

Area of rectangle is base*height

Area of triangle is $\frac{1}{2}$ * base * height

Area of circle is 3.14159 * radius *radius

Example:

Enter number of objects: 4

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

Enter type of object 2: s enter length of square: 10.5

Enter type of object 3: r enter height of rectangle: 10 enter width of rectangle: 5

Enter type of object 4: t enter height of triangle: 5 enter base of triangle: 5

The largest is object 2 of type s with area 110.25

A1 Geometric Objects Test Cases

The exact wording of prompts is not important. Only the required number of inputs should be used

First Round Testing

Test 1: *Enter number of objects: 4*

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

Enter type of object 2: s enter length of square: 10

Enter type of object 3: r enter height of rectangle: 10 enter width of rectangle: 5

Enter type of object 4: t enter height of triangle: 5 enter base of triangle: 5

The largest is object 2 of type s with area 100

Test 2: *Enter number of objects: 1*

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

The largest is object 1 of type t with area 50

Test 3: *Enter number of objects: 6*

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

Enter type of object 2: s enter length of square: 10

Enter type of object 3: c enter radius of circle: 20

Enter type of object 4: c enter radius of circle: 20

Enter type of object 5: t enter height of triangle: 5 enter base of triangle: 5

Enter type of object 6: t enter height of triangle: 5 enter base of triangle: 5

The largest is object 3 (or 4) of type c with area 1256.64 (don't worry about decimals)

Second Round Testing: all of the above tests plus the following

Test 4: *Enter number of objects: 4*

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

Enter type of object 2: s enter length of square: 10

Enter type of object 3: r enter height of rectangle: 10 enter width of rectangle: 9

Enter type of object 4: t enter height of triangle: 20 enter base of triangle: 11

The largest is object 4 of type t with area 110

Test 5: *Enter number of objects: 4*

Enter type of object 1: t enter height of triangle: 10 enter base of triangle: 10

Enter type of object 2: s enter length of square: 10

Enter type of object 3: r enter height of rectangle: 10 enter width of rectangle: 11

Enter type of object 4: t enter height of triangle: 5 enter base of triangle: 5

The largest is object 3 of type r with area 110

A1GeoArea Code

```
// GeoAreaA.cpp : This project reads upto ten descriptions of geometric
// figures. It then finds the object with the largest surface area.
// It will print out the ordinal number (e.g. 2 for second), the
// type of figure (e.g. triangle), and the area of the figure.
// The input for each object will be a char (s for square, r for
// rectangle, c for circle, t for triangle) and the necessary
// lengths.
// Assume that there is at least one object and that the input is
// syntactically correct. For example only one length is given for
// a circle's radius.

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;
using namespace System;
#define MAX 10

int main(array<System::String ^> ^args)
{
    char *type = new char[MAX];
    float length1, length2, float area[MAX];
    int numGeoObj, int i;
    cout<< "\nEnter number of objects: "; cin>> numGeoObj;
    for(i=0; i< numGeoObj; i++)
    {
        cout<< "\nEnter type of object: ";
        cin>> type[i];
        if(type[i]== 's' || type[i] == 'S')
        {
            cout<< "\nEnter one length:";
            cin>> length1;
            area[i] = length1 * length1;
        }
        if(type[i]== 'r' || type[i] == 'R')
        {
            cout<< "\nEnter height:"; cin>> length1;
            cout<< "\nEnter length:"; cin>> length2;
            area[i] = length1 * length2;
        }
        if(type[i]== 't' || type[i] == 'T')
        {
            cout<< "\nEnter height of triangle:"; cin>> length1;
            cout<< "\nEnter length of base:"; cin>> length2;
            area[i] = .5 * length1 * length2;
        }
        if(type[i]== 'c' || type[i] == 'C')
        {
            cout<< "\nEnter radius of circle:"; cin>> length1;
            area[i] = 3.14159 * length1 * length1;
        }
    }

    int maxarea = area[0];
```

```
int maxobj = 0;

for(i=1; i<numGeoObj; i++)
{
    if (area[i] >maxarea)
    {
        maxobj = i;
        maxarea = area[i];
    }
}

cout<< "\nThe largest figure is number "<< maxobj+1 << " of type "
      << type[maxobj] << " with an area of " << area[maxobj];
return 0;
}
```

2 Advanced — Optimal Chocolate Purchase

We have a certain amount of money that we wish to use to buy chocolate bars. There are three sizes of chocolate bars, and we wish to buy a combination of these three sizes that maximizes the amount of chocolate without exceeding the amount of money that we have. Write a program that takes an amount of money in cents (so that you only need integer values), the cost in cents of the three sizes, and the three sizes in ounces. The program must then display the number of each size, the total weight, and the total cost for an optimal purchase. You may assume that all inputs are positive integers.

Example 1:

```
Enter the amount of money (in cents): 575
Enter the cost of the first candy bar (in cents): 50
Enter the size of the first candy bar (in ounces): 3
Enter the cost of the second candy bar (in cents): 100
Enter the size of the second candy bar (in ounces): 7
Enter the cost of the third candy bar (in cents): 200
Enter the size of the third candy bar (in ounces): 17
```

```
Buy 1 of the first, 1 of the second, and 2 of the third.
Total size: 44 ounces
Total cost: 550 cents
```

Example 2:

```
Enter the amount of money (in cents): 1100
Enter the cost of the first candy bar (in cents): 112
Enter the size of the first candy bar (in ounces): 17
Enter the cost of the second candy bar (in cents): 85
Enter the size of the second candy bar (in ounces): 13
Enter the cost of the third candy bar (in cents): 46
Enter the size of the third candy bar (in ounces): 7
```

```
Buy 1 of the first, 10 of the second, and 3 of the third.
Total size: 168 ounces
Total cost: 1100 cents
```

2 Advanced — Optimal Chocolate Purchase Test Cases

Test Case 1:

Enter the amount of money (in cents): 1189
Enter the cost of the first candy bar (in cents): 99
Enter the size of the first candy bar (in ounces): 9
Enter the cost of the second candy bar (in cents): 119
Enter the size of the second candy bar (in ounces): 11
Enter the cost of the third candy bar (in cents): 204
Enter the size of the third candy bar (in ounces): 19

Buy 1 of the first, 4 of the second, and 3 of the third.
Total size: 110 ounces
Total cost: 1187 cents

Test Case 2:

Enter the amount of money (in cents): 1100
Enter the cost of the first candy bar (in cents): 112
Enter the size of the first candy bar (in ounces): 17
Enter the cost of the second candy bar (in cents): 85
Enter the size of the second candy bar (in ounces): 13
Enter the cost of the third candy bar (in cents): 46
Enter the size of the third candy bar (in ounces): 7

Buy 1 of the first, 10 of the second, and 3 of the third.
Total size: 168 ounces
Total cost: 1100 cents

Second Submission: Do the above tests, plus the following:

Test Case 3:

Enter the amount of money (in cents): 575
Enter the cost of the first candy bar (in cents): 59
Enter the size of the first candy bar (in ounces): 5
Enter the cost of the second candy bar (in cents): 119
Enter the size of the second candy bar (in ounces): 11
Enter the cost of the third candy bar (in cents): 49
Enter the size of the third candy bar (in ounces): 3

Buy 0 of the first, 4 of the second, and 2 of the third.
Total size: 50 ounces
Total cost: 574 cents

```

/* Advanced 2 - Optimal Chocolate Purchase
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.ChocolateAdvanced
{
    class Chocolate
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the amount of money (in cents): ");
            int money = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the first candy bar (in cents): ");
            int cost1 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the size of the first candy bar (in ounces): ");
            int size1 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the second candy bar (in cents): ");
            int cost2 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the size of the second candy bar (in ounces): ");
            int size2 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the cost of the third candy bar (in cents): ");
            int cost3 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter the size of the third candy bar (in ounces): ");
            int size3 = Int32.Parse(Console.ReadLine());
            Console.WriteLine();
            int num1 = 0;
            int num2 = 0;
            int num3 = 0;
            int totalSize = 0;
            for (int i = 0; i <= money / cost1; i++)
            {
                for (int j = 0; j <= (money - i * cost1) / cost2; j++)
                {
                    int k = (money - i * cost1 - j * cost2) / cost3;
                    if (i * size1 + j * size2 + k * size3 >= totalSize)
                    {
                        num1 = i;
                        num2 = j;
                        num3 = k;
                        totalSize = num1 * size1 + num2 * size2 + num3 * size3;
                    }
                }
            }
            Console.WriteLine("Buy " + num1 + " of the first, " + num2 +
                " of the second, and " + num3 + " of the third.");
            Console.WriteLine("Total size: " + totalSize + " ounces");
            Console.WriteLine("Total cost: " +
                (num1 * cost1 + num2 * cost2 + num3 * cost3) + " cents");
            Console.ReadLine();
        }
    }
}

```

A3 Path Problem Statement

This program accepts information about paths between nodes. Each node is represented by an integer between 1 and 20. There will be N data inputs. Each data input has a starting node, a destination node, and the distance between the nodes. Each path is one way.

Sample Graph:

```
Enter start node: 1 end node: 4 distance: 5  
Enter start node: 1 end node: 6 distance: 10  
Enter start node: 4 end node: 2 distance: 3  
Enter start node: 6 end node: 3 distance: 5  
Enter start node: 3 end node: 2 distance: 5
```

The program will accept a path (that is a list of consecutive nodes) and respond whether the path exists and the length of the path. All data entries for locations will be integers between 1 and 20. In this example, the path of

1 6 3 2 would be a correct path with a total weight of 20. On the other hand, neither 1 3 2 nor 6 1 4 2 would be a correct path.

The number of data inputs will be between 1 and 10. The number of nodes in the path will be between 1 and 10.

Example 1:

```
Enter number of data inputs: 7  
Enter start node: 1 end node: 4 distance: 5  
Enter start node: 1 end node: 6 distance: 10  
Enter start node: 4 end node: 2 distance: 3  
Enter start node: 6 end node: 1 distance: 5  
Enter start node: 3 end node: 2 distance: 5  
Enter start node: 4 end node: 6 distance: 7  
Enter start node: 6 end node: 3 distance: 5  
  
Enter number of nodes in path: 3  
Enter path start: 1 enter next: 4 enter next: 2  
Path is correct. Total distance is 8
```

Example 2:

Same input for graph

```
Enter number of nodes in path: 6  
Enter path start: 1 enter next: 6 enter next: 1  
enter next: 6 enter next: 4 enter next: 1  
Path is incorrect.
```

A3 Path Test Cases

The exact format of the prompts is not important.

First Round

Test 1:

```
Enter number of data inputs: 7
Enter start node: 1 end node: 4 distance: 5
Enter start node: 1 end node: 6 distance: 10
Enter start node: 4 end node: 2 distance: 3
Enter start node: 6 end node: 1 distance: 5
Enter start node: 3 end node: 2 distance: 5
Enter start node: 4 end node: 6 distance: 7
Enter start node: 6 end node: 3 distance: 5

Enter number of nodes in path: 3
Enter path start: 1 enter next: 4 enter next: 2
Path is correct. Total distance is 8
```

Test 2: Same input for graph

```
Enter number of nodes in path: 6
Enter path start: 1 enter next: 6 enter next: 1
enter next: 6 enter next: 4 enter next: 1
Path is incorrect.
```

Second Round: repeat the first two tests and the following tests

Test 3:

```
Enter number of data inputs: 7
Enter start node: 1 end node: 2 distance: 10
Enter start node: 1 end node: 3 distance: 10
Enter start node: 4 end node: 2 distance: 6
Enter start node: 4 end node: 1 distance: 2
Enter start node: 3 end node: 2 distance: 9
Enter start node: 3 end node: 6 distance: 4
Enter start node: 6 end node: 3 distance: 6

Enter number of nodes in path: 6
Enter path start: 4 enter next: 1 enter next: 3
enter next: 6 enter next: 3 enter next: 2
Path is correct. Total distance is 31
```

Test 4: Use same graph as test 3

```
Enter number of nodes in path: 1
Enter path start: 6
Path is correct. Total distance is 0
```

APath Code

```
// PathA.cpp : This program accepts information about paths between
// destinations. Each destination is represented by an
// integer between 1 and 20. There will be N data inputs. Each
// data input has a starting node, a destination node,
// and the distance between points. Each path is one way. The
// program will accept a path (that is a list of nodes
// and respond whether the path exists and the length of the path.
// All data entries for node will be integers
// between 1 and 20
#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;
using namespace System;
#define MAX 20

int main(array<System::String ^> ^args)
{
    int road[MAX][MAX], numRoads, numSteps, totalDist;
    int start, end, distance;
    int loc1, loc2, totalDis, i, j;

    // clear road array

    for(i=0; i<20; i++)
    {
        for(j=0; j<20; j++)
        {
            road[i][j] = 0;
        }
    }
    // enter road data
    cout<< "\nEnter number of roads: "; cin>> numRoads;
    for(i=0; i<numRoads; i++)
    {
        cout<< "\nEnter start: "; cin>> start;
        cout<< "\nEnter end: "; cin>> end;
        cout<< "\nEnter distance: "; cin>> distance;
        road[start-1][end-1]=distance;
    }
    // enter path to check
    cout<< "\nEnter number of nodes in path: "; cin>> numSteps;
    cout<< "\nEnter start node: "; cin>> loc1;
    totalDis = 0;
    for(i=1; i<numSteps; i++)
    {
        cout<< "\nEnter next node: "; cin>> loc2;
        distance = road[loc1-1][loc2-1];
        if (distance == 0)
        {
            cout<< "\nError in path: road missing from "<< loc1 <<" to
            "<< loc2;
            return 0;
        }
    }
}
```

```
        totalDis = totalDis + distance;
        loc1 = loc2;
    }
    cout<< "\nPath successful!  total distance is "<<totalDis;
    return 0;
}
```

4 Advanced — Approximating π

The mathematical constant π can be approximated as follows. For each nonnegative integer n , let

$$\pi_{0,n} = 4 \sum_{i=0}^n \frac{(-1)^i}{2i+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \cdots + \frac{4(-1)^n}{2n+1}$$

and for each positive integer m and each nonnegative integer n , let

$$\pi_{m,n} = \frac{\pi_{m-1,n} + \pi_{m-1,n+1}}{2}.$$

For example,

$$\pi_{1,2} = \frac{\pi_{0,2} + \pi_{0,3}}{2} = \frac{\left(\frac{4}{1} - \frac{4}{3} + \frac{4}{5}\right) + \left(\frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7}\right)}{2}.$$

As m and n increase, $\pi_{m,n}$ converges to π . Write a program that takes as input nonnegative integers m and n and produces as output $\pi_{m,n}$. Note that in order to compute $\pi_{m,n}$, you will need values such as $\pi_{m-1,n+1}$, $\pi_{m-2,n+2}$, etc. You may assume that $m+n < 100$.

Example 1:

```
Enter m: 1
Enter n: 0
Pi is approximately 3.33333333333333
```

Example 2:

```
Enter m: 0
Enter n: 2
Pi is approximately 3.46666666666667
```

Example 3:

```
Enter m: 4
Enter n: 5
Pi is approximately 3.14154589820225
```

Note: Your answers may not be identical to those above, but should agree when rounded to 4 digits (i.e., 3 digits to the right of the decimal point).

4 Advanced — Approximating π Test Cases

The answers produced must match when rounded to 4 digits (i.e., 3 digits to the right of the decimal point).

Test Case 1:

```
Enter m: 0
Enter n: 0
Pi is approximately 4
```

Test Case 2:

```
Enter m: 3
Enter n: 2
Pi is approximately 3.14343434343434
```

Test Case 3:

```
Enter m: 99
Enter n: 0
Pi is approximately 3.14159265358979
```

Second Submission: Do the above tests, plus the following:

Test Case 4:

```
Enter m: 50
Enter n: 49
Pi is approximately 3.14159265358979
```

```

/* Advanced 4 - Approximating Pi
 * This solution uses a singly-dimensioned array to store a single row
 * of values. As the next row is computed, the current row is
 * overwritten.
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.PiAdvanced
{
    class Pi
    {
        static void Main(string[] args)
        {
            Console.Write("Enter m: ");
            int m = Int32.Parse(Console.ReadLine());
            Console.Write("Enter n: ");
            int n = Int32.Parse(Console.ReadLine());
            double[] pi = new double[m + n + 1];
            pi[0] = 4.0;
            double numerator = -4.0;
            for (int j = 1; j <= m + n; j++)
            {
                pi[j] = pi[j - 1] + numerator / (2 * j + 1);
                numerator *= -1;
            }
            for (int i = 1; i <= m; i++)
            {
                // After the first row is computed, we don't need the
                // first n values in any row.
                for (int j = n; j <= m + n - i; j++)
                {
                    pi[j] = (pi[j] + pi[j + 1]) / 2.0;
                }
            }
            Console.WriteLine("Pi is approximately " + pi[n]);
            Console.ReadLine();
        }
    }
}

```

A5 Tiling Problem Statement

This program is to calculate how many tiles are needed to tile a floor. There will be two different sizes. The tiles are square. The size of the tiles is read in and the price for a box of 100 is also entered. All lengths are in inches. Only whole boxes of tiles can be bought.

Tiles can be used whole or a part of a tile can be used. Only one usable piece can be cut from a tile. That is, if a piece is cut from a tile, the rest of the tile must be thrown away.

The program accepts the length and width of the room and determines which tile size is most economical and returns how many whole tiles are used and how many part tiles are used and the total price.

Example 1:

```
Enter the length of tile 1: 10 Enter the cost of a box of 100: 70  
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80  
Enter the length of the room in inches: 120  
Enter the width of the room in inches: 240  
The lower cost is using tile 2. 200 whole tiles and 0 part tiles are  
necessary. The cost is $160.
```

Example 2:

```
Enter the length of tile 1: 10 Enter the cost of a box of 100: 70  
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80  
Enter the length of the room in inches: 95  
Enter the width of the room in inches: 190  
The lower cost is using tile 1. 171 whole tiles and 19 part tiles are  
necessary. The cost is $140.
```

A5 Tiling Test Cases

The exact format of the prompts is unimportant

First Round

Test 1:

Enter the length of tile 1: 10 Enter the cost of a box of 100: 70
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80
Enter the length of the room in inches: 120
Enter the width of the room in inches: 240
The cost is less with tile 2. The total number of whole tiles is 200 and part tiles is 0. The cost is \$160

Test 2

Enter the length of tile 1: 10 Enter the cost of a box of 100: 70
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80
Enter the length of the room in inches: 140
Enter the width of the room in inches: 230
The cost is less with tile 2. The total number of whole tiles is 209 and part tiles is 31. The cost is \$240

Test 3

Enter the length of tile 1: 10 Enter the cost of a box of 100: 70
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80
Enter the length of the room in inches: 100
Enter the width of the room in inches: 200
The cost is less with tile 1. The total number of whole tiles is 200 and part tiles is 0. The cost is \$140

Second round: repeat the above tests and do the additional test below

Test 4

Enter the length of tile 1: 10 Enter the cost of a box of 100: 70
Enter the length of tile 2: 12 Enter the cost of a box of 100: 80
Enter the length of the room in inches: 95
Enter the width of the room in inches: 190
The lower cost is using tile 1. 171 whole tiles and 19 part tiles are necessary. The cost is \$140.

A5Tiling Code

```
// TilingA.cpp : This program is to calculate how many tiles are needed to
// tile a floor. There will be two different sizes. The tiles are square.
// The size of the tiles is read in and the price for a box of 100 is
// entered.
// Tiles can be used whole or a part of a tile can be used. Only one
// usable piece can be cut from a tile. That is, if a piece is cut from a
// tile, the rest of the tile must be thrown away. The program accepts
// the length and width of the room and determines which tile size is most
// economical and returns how many whole tiles are used and how many part
// tiles are used and the total price

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    float width, length;
    int tile1size, tile2size, boxes1, boxes2, cost1, cost2;
    float tile1cost, tile2cost;
    int whole1tiles, part1tiles, tile1width, tile1length;
    int extrawide1, extralength1;

    int whole2tiles, part2tiles, tile2width, tile2length;
    int extrawide2, extralength2;

    cout<<"\nEnter the width of tile 1 in inches: "; cin >> tile1size; cout
    <<"\nEnter the cost of a box of 100 tile 1: "; cin >> tile1cost;
    cout<<"\nEnter the width of tile 2 in inches: "; cin >> tile2size; cout
    <<"\nEnter the cost of a box of 100 tile 2: "; cin >> tile2cost;
    cout<<"\nEnter the width of the floor in inches: "; cin >> width;
    cout<<"\nEnter the length of the floor in inches: "; cin >> length;

    // calculate for tile 1

    tile1width = (int) width/tile1size;
    tile1length = (int) length/tile1size;
    part1tiles = 0;
    whole1tiles = tile1width * tile1length;
    extrawide1 = width - tile1size*tile1width;
    extralength1 = length - tile1size*tile1length;

    if (extrawide1 > 0) {part1tiles = tile1length;}
    if (extralength1 > 0) {part1tiles = part1tiles + tile1width;}
    if (extrawide1 > 0 && extralength1 > 0) {part1tiles++;}

    cout<<"\nThe number of whole tiles for Tile One is " << whole1tiles
    <<" and the number of part tiles is " << part1tiles;
    boxes1 = (int) (whole1tiles + part1tiles)/100;
    if ((whole1tiles+part1tiles) - 100* boxes1 > 0) {boxes1++;}
    cost1 = boxes1 * tile1cost;
```

```

// calculate for tile 2

tile2width = (int) width/tile2size; tile2length = (int)
    length/tile2size;
part2tiles = 0;
whole2tiles = tile2width * tile2length;
extrawide2 = width - tile2size*tile2width;
extralength2 = length - tile2size*tile2length;

if (extrawide2 > 0) {part2tiles = tile2length;}
if (extralength2 > 0) {part2tiles = part2tiles + tile2width;}
if (extrawide2 > 0 && extralength2 > 0) {part2tiles++;}

cout<< "\nThe number of wholes tiles for Tile Two is " << whole2tiles
    << " and the number of part tiles is " << part2tiles;
boxes2 = (int)(whole2tiles + part2tiles)/100;
if ((whole2tiles+part2tiles) - 100* boxes2 > 0) {boxes2++;}
cost2 = boxes2 * tile2cost;

if(cost1 < cost2)
{cout<< "\n Tile One is cheaper.  It will take "<<boxes1
    << " boxes of tile for a total cost of $" << cost1 ;}
else
{cout<< "\n Tile Two is cheaper.  It will take "<<boxes2
    << " boxes of tile for a total cost of $" << cost2 ;}

return 0;
}

```

6 Advanced — Subsequence Counting

Write a program that takes a text string and a target string as input, and outputs the number of occurrences of the target string as a subsequence of the text string. The subsequences need to occur in the same order in both strings, but do not necessarily need to be contiguous in the text string; for example, the target “ab” occurs 3 times in the text “abab”: abab, abab, and abab. Both strings must be input exactly as the strings to be used, with nothing artificially separating the characters. You may assume that both strings are nonempty, that the text is no longer than 20 characters, and that the target is no longer than 5 characters.

Example 1:

```
Enter text string: abab
Enter target string: ab
The target occurs 3 times.
```

Example 2:

```
Enter text string: Mississippi
Enter target string: isi
The target occurs 14 times.
```

Example 3:

```
Enter text string: structure
Enter target string: data
The target occurs 0 times.
```

6 Advanced — Subsequence Counting Test Cases

Test Case 1:

Enter text string: Mississippi
Enter target string: isi
The target occurs 14 times.

Test Case 2:

Enter text string: aaaabaaaabaaaabaaaab
Enter target string: aaaaa
The target occurs 4368 times.

Test Case 3:

Enter text string: banana
Enter target string: ab
The target occurs 0 times.

Second Submission: Do the above tests, plus the following:

Test Case 4:

Enter text string: abcdeabcdeabcdeabcde
Enter target string: abcde
The target occurs 56 times.

```

/* Advanced 6 - Subsequence Counting
*
* This problem can be solved using 5 nested loops, or more generally
* using recursion. However, the following solution is general (i.e.,
* it doesn't depend on the maximum length of either string) and much
* more efficient. It computes a 2-dimensional array count[,] such that
* count[i, j] gives the number of occurrences of the first i characters
* of the target in the first j characters of the text. The trickiest
* part is determining this value when i = 0. A value of 1 allows a
* uniform computation for the remainder of the array.
*/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Contest2009.SubsequenceAdvanced
{
    class Subsequence
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter text string: ");
            string text = Console.ReadLine();
            Console.WriteLine("Enter target string: ");
            string target = Console.ReadLine();
            int[,] count = new int[target.Length + 1, text.Length + 1];
            for (int j = 0; j <= text.Length; j++)
            {
                count[0, j] = 1;
            }
            for (int i = 1; i <= target.Length; i++)
            {
                count[i, 0] = 0;
                for (int j = 1; j <= text.Length; j++)
                {
                    if (target[i - 1] == text[j - 1])
                    {
                        count[i, j] = count[i, j - 1] + count[i - 1, j - 1];
                    }
                    else
                    {
                        count[i, j] = count[i, j - 1];
                    }
                }
            }
            Console.WriteLine("The target occurs " +
                count[target.Length, text.Length] + " times.");
            Console.ReadLine();
        }
    }
}

```