Contents

Chapter 1

Rubric

Question	Points
Question 1	10
Question 2	10
Question 3	10
Question 4	10
Question 5	10
`tugOfWar`	
Test Cases	1×15
Compilation	10
`tugOfWar` Total	25
`footRace`	
Test Cases	1×15
Compilation	10
`footRace` Total	25
Total	100

Chapter 2

Metadata

2.1 Submitted Files

handin.time

```
11/13/2019 18:12:54 fsandhu: csce310h0mework04part02.cpp          - OK
11/13/2019 18:12:56 fsandhu: csce310h0mework04part02.h    - OK
11/13/2019 18:40:42 fsandhu: csce310h0mework04part02.cpp          - OK
11/13/2019 18:50:14 fsandhu: csce310h0mework04part01.cpp          - OK
11/13/2019 18:50:16 fsandhu: csce310h0mework04part01.h    - OK
11/13/2019 18:51:49 fsandhu: csce310h0mework04part01.cpp          - OK
11/18/2019 18:13:39 fsandhu: csce310h0mework04part01.cpp          - OK
11/18/2019 18:17:55 fsandhu: csce310h0mework04part01.cpp          - OK
11/18/2019 18:21:24 fsandhu: csce310h0mework04part01.cpp          - OK
11/18/2019 18:35:37 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 18:38:00 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 18:39:21 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 18:42:07 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 19:07:34 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 19:13:57 fsandhu: csce310h0mework04part02.cpp          - OK
11/18/2019 19:24:23 fsandhu: csce310h0mework04part03.cpp          - OK
11/18/2019 19:24:26 fsandhu: csce310h0mework04part03.h    - OK
11/18/2019 19:24:50 fsandhu: csce310h0mework04part03.cpp          - OK
11/18/2019 19:26:45 fsandhu: csce310h0mework04part03.cpp          - OK
11/19/2019 13:46:14 fsandhu: csce310h0mework04part02.cpp          - OK
11/19/2019 13:47:55 fsandhu: csce310h0mework04part02.cpp          - OK
11/19/2019 14:05:17 fsandhu: csce310h0mework04part02.cpp          - OK
11/19/2019 14:06:28 fsandhu: csce310h0mework04part02.cpp          - OK
11/19/2019 15:43:34 fsandhu: csce310h0mework04part01.cpp          - OK
11/19/2019 15:43:36 fsandhu: csce310h0mework04part01.h    - OK
11/19/2019 15:43:38 fsandhu: csce310h0mework04part02.cpp          - OK
11/19/2019 15:43:40 fsandhu: csce310h0mework04part03.cpp          - OK
11/19/2019 15:43:43 fsandhu: csce310h0mework04part02.h    - OK
11/19/2019 15:43:46 fsandhu: csce310h0mework04part03.h    - OK
11/21/2019 22:02:31 fsandhu: fsandhu_hw04.pdf     - 1 day late
```

2.2 webgrader Runs

webgrader.time

```
2019-11-13T18:13:09-0600 76.84.50.181  fsandhu 004
2019-11-13T18:40:48-0600 76.84.50.181  fsandhu 004
2019-11-13T18:50:19-0600 76.84.50.181  fsandhu 004
2019-11-13T18:51:51-0600 76.84.50.181  fsandhu 004
2019-11-18T18:13:52-0600 10.43.83.198  fsandhu 004
2019-11-18T18:18:05-0600 10.43.83.198  fsandhu 004
```

```
2019-11-18T18:21:25-0600 10.43.83.198  fsandhu 004
2019-11-18T18:22:07-0600 10.43.83.198  fsandhu 004
2019-11-18T18:35:40-0600 10.43.83.198  fsandhu 004
2019-11-18T18:35:46-0600 10.43.83.198  fsandhu 004
2019-11-18T18:38:02-0600 10.43.83.198  fsandhu 004
2019-11-18T18:38:07-0600 10.43.83.198  fsandhu 004
2019-11-18T18:39:24-0600 10.43.83.198  fsandhu 004
2019-11-18T18:39:29-0600 10.43.83.198  fsandhu 004
2019-11-18T18:42:10-0600 10.43.83.198  fsandhu 004
2019-11-18T18:42:16-0600 10.43.83.198  fsandhu 004
2019-11-18T19:07:37-0600 10.43.83.198  fsandhu 004
2019-11-18T19:07:43-0600 10.43.83.198  fsandhu 004
2019-11-18T19:14:00-0600 10.43.83.198  fsandhu 004
2019-11-18T19:24:28-0600 10.43.83.198  fsandhu 004
2019-11-18T19:24:52-0600 10.43.83.198  fsandhu 004
2019-11-18T19:25:16-0600 10.43.83.198  fsandhu 004
2019-11-18T19:26:46-0600 10.43.83.198  fsandhu 004
2019-11-18T19:26:55-0600 10.43.83.198  fsandhu 004
2019-11-19T13:23:54-0600 10.43.73.242  fsandhu 004
2019-11-19T13:46:21-0600 10.43.73.242  fsandhu 004
2019-11-19T13:46:29-0600 10.43.73.242  fsandhu 004
2019-11-19T13:48:01-0600 10.43.73.242  fsandhu 004
2019-11-19T13:48:07-0600 10.43.73.242  fsandhu 004
2019-11-19T14:05:18-0600 10.43.73.242  fsandhu 004
2019-11-19T14:05:56-0600 10.43.73.242  fsandhu 004
2019-11-19T14:06:30-0600 10.43.73.242  fsandhu 004
2019-11-19T14:06:36-0600 10.43.73.242  fsandhu 004
2019-11-19T15:36:30-0600 10.43.73.242  fsandhu 004
2019-11-19T15:43:52-0600 10.43.73.242  fsandhu 004
2019-11-19T15:43:58-0600 10.43.73.242  fsandhu 004
2019-11-21T22:02:49-0600 76.84.50.181  fsandhu 004
2019-12-15T19:53:01-0600 76.84.219.87  fsandhu 004
```

2.3 diffs

submission.diffs

Chapter 3

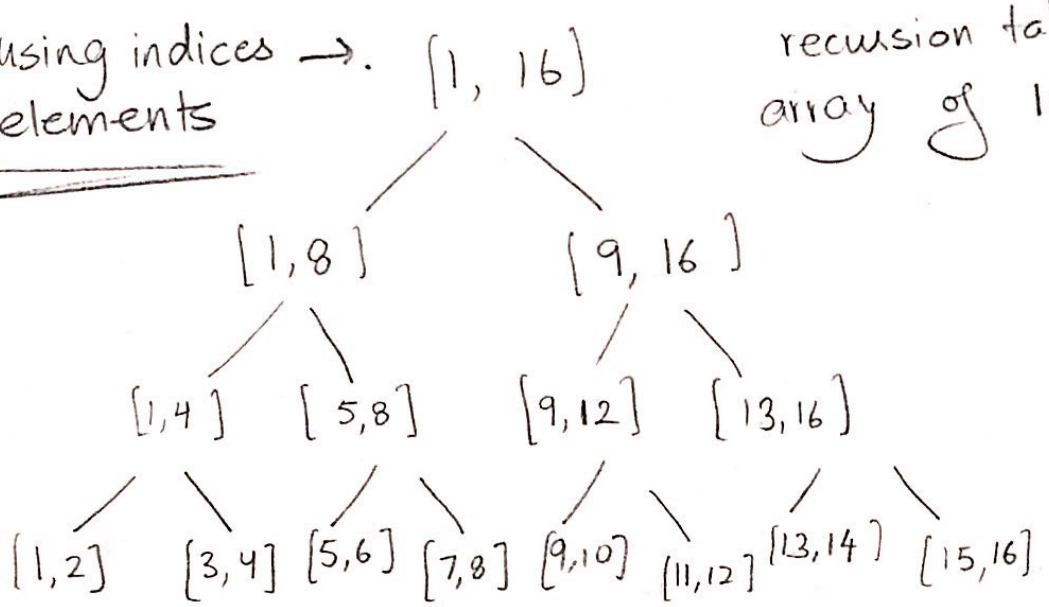
Written Exercises

Assignment #4

Fateh Sandhu (17286643)

(1) using indices of elements →

recursion table for array of 16 elements



memoization is ineffective in speeding up a good divide-and-conquer algorithm because no computation is repeated in a good divide and conquer algorithm and same number of computations will be done in memoization.

So dynamic programming & recursion take same time complexity.

Q2)

Coin row problem

recurrence formula

$$F(n) = \max(c_n + F(n-2), F(n-1)) \quad \text{for } n > 1$$

$$F(0) = 0 \qquad F(1) = c_1$$

$$F(2) = \max(c_2 + F(0), F(1))$$

$$F(3) = \max(c_3 + F(1), F(2))$$

$F(4) = \max(c_4 + F(2), F(3))$

compute F at each stage and substitute computations will give us an exponential number of comparison

at each step, we are making one of two possible choices for n steps; in each recursion so total choices / comparisons made will be 2^n which an exponential running time . → $O(2^n)$

b) Exhaustive search will result in computing all possible combinations and then choosing the maximum value out of those. Factorial running time is at least more than exponential running time so exhaustive search will be at least exponential in running time

$O(n!) \geq O(2^n)$.

Q3)

Using recursion, the number of function calls will be almost exponential if we want to compute $C(n, n/2)$ assuming n is even.

$$C(n, n/2) = \frac{n!}{(n/2)!\,(n-n/2)!}$$

$$= \frac{n!}{(n/2)!\,(n/2)!} = \frac{n!}{(n/2!)^2}$$

Stirling's formula

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

This is also known as Stirling's approximation formula

Computation of $C(n, n/2)$ will also have the maximum number of recurrence calls.

applying Stirling's formula on $\frac{n!}{(n/2!)^2}$ $\approx 2^{n/2}$

So it will be $O(2^n)$ and the time complexity will be exponential

b) using memoization, we can create a $n \times k$ table and store each computation of $C(n,k)$ instead of recursion.

By memoization, denoting rows as i and columns as j

$$\text{matrix}(i,j) = \text{matrix}(i-1)(j-1) + \text{matrix}(i-1)(j).$$

If we are only building a matrix until i & j then bottom right element will be our answer and we will manually fill in all base cases. Algorithm will be $O(n^2)$

(04) We can solve this problem by dynamic programming. Using memoization, this problem can be solved in $O(mn)$ time because we just have to fill in a matrix of size $m \times n$.

Proceduce;

Example:

FOOTBALL & BALL

	ϕ	F	O	O	T	B	A	L	L
ϕ	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	1	2	2	2
L	0	0	0	0	0	1	2	3	3
L	0	0	0	0	0	1	2	3	4

4 → longest substring length

largest common substring is BALL

At each index in matrix; computation is as follows;

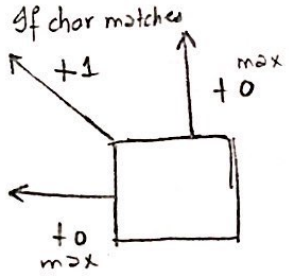
$$matrix[i][j] = \begin{cases} \text{if } word1.charAT[j] == word2.charAT[i] \\ \text{then } matrix[i-1][j-1] + 1 \\ \text{else} \\ \max(matrix[i-1][j], matrix[i][j-1]) \end{cases}$$

filling up the matrix will take at most 2mn comparisons as each element makes one comparison for the character matching and if characters don't match, then one more comparison for choosing the max.

Time complexity = O(mn)

Space Complexity = O(mn)

general idea for each element of matrix



The first column & row are base cases and can be filled in with zeroes

05) Counting heads.

This problem can be solved with dynamic programming. For this we create a mn memoization matrix when we have m coins and need n heads.

Input is a vector or array containing probability of a head in each coin.

Each element in matrix is probabity of j' heads in i tosses

Procedure:

create a mxn memoization matrix and fill in row and columns for base cases.

Example: 3 heads in 6 tosses.

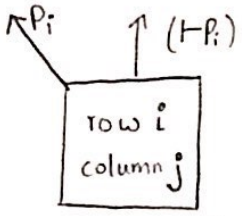
arrayH: 0.7 , 0.1 , 0.2 , 0.7 , 0.1 , 0.9

# of heads

First n coins	0	1	2	3
0	1	0	0	0
1	$(1-H_1)$	H_1	0	0
2	$(1-H_1)(1-H_2)$	$H_1(1-H_2)$ $+H_2(1-H_1)$	H_1H_2	0
3	⋮	⋮	and so on.	
4	⋮	⋮		
5	⋮	⋮		
6	$\prod_{i=1}^{6}(1-H_i)$			

Fill in the whole matrix

general idea for each element



i → first n coins

j → # of heads.

if j > i
set element = 0

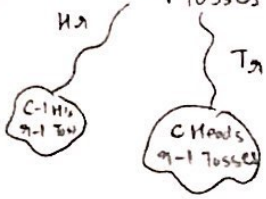
else

i-j (probability of tails)
x j (probability of heads)

$P(C,n) = P(H_n) + P(C-1, n-1)$

OR

$P(T_n) + P(C, n-1)$



The answer will be in the bottom right element of memoization matrix which will give the exact probability of j heads in i tosses.

ie matrix(m)(n)

consider every combination and compute it and add them all up.

for example: if we have 2 tails and 1 head for 3 coin tosses

compute $H_1T_2T_3 + T_1H_2T_3 + T_1T_2H_3$

where H_i → Heads on i^{th} coin
and T_i → Tails on i^{th} coin.

Probabilities can be computed from the arroy passed in as an iput

* Dynamic Programming will save time for some comparisons which can directly be looked up in the memoization matrix

In worst case this will have a $O(n^2)$ running complexity

Chapter 4

Programming Exercises

4.1 csce310h0mework04part01

4.1.1 Test 01

diff

part01test01.diff

Input

part01test01.input

```
47 57 98 29 49 58 31 28 39 89 28 60 67 85 82
```

Submission Output

part01test01.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test01.solution

```
Team 01
28 89 39 28 58 49 29 57 47
Team 02
98 31 60 67 85 82
The two teams can differ by at most 1 pound(s).
```

stderr

part01test01.err

4.1.2 Test 02

diff

part01test02.diff

Input

part01test02.input

```
47 74 83 36 81 54 22 10 17 46 99 31 31 55 62 51 49 85 64 87 50 26 91 56 15 54 84 39 30
```

Submission Output

part01test02.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test02.solution

```
Team 01
49 62 55 31 31 99 46 17 54 81 36 83 74 47
Team 02
22 10 51 85 64 87 50 26 91 56 15 54 84 39 30
The two teams can differ by at most 1 pound(s).
```

stderr

part01test02.err

4.1.3 Test 03

diff

part01test03.diff

Input

part01test03.input

```
35 41 24 26 63 27 54 28 98 24 48 87 30 61 30 68 68 62 51 11 49 56 39 92 10 24 93 74 10
    20
```

Submission Output

part01test03.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test03.solution

```
Team 01
68 68 61 87 48 98 28 54 27 63 24 41 35
Team 02
26 24 30 30 62 51 11 49 56 39 92 10 24 93 74 10 20
The two teams can differ by at most 1 pound(s).
```

stderr

part01test03.err

4.1.4 Test 04

diff

part01test04.diff

Input

part01test04.input

```
90 43 41 45 64 17 61 94 84 14 13 61 73 51 63 95 12 32 15 54 74 88
```

Submission Output

part01test04.output

```
The two teams can differ by at most 0 pound(s).
```

Solution Output

part01test04.solution

```
Team 01
73 61 84 94 61 45 41 43 90
Team 02
64 17 14 13 51 63 95 12 32 15 54 74 88
The two teams can differ by at most 0 pound(s).
```

stderr

part01test04.err

4.1.5 Test 05

diff

part01test05.diff

Input

part01test05.input

```
34 20 51 69 53 67 19 90
```

Submission Output

part01test05.output

```
The two teams can differ by at most 7 pound(s).
```

Solution Output

part01test05.solution

```
Team 01
90 19 69 20
Team 02
34 51 53 67
The two teams can differ by at most 7 pound(s).
```

stderr

part01test05.err

4.1.6 Test 06

diff

part01test06.diff

Input

part01test06.input

```
43 15 46 84 14 77 53 29 59 31 55 58 21
```

Submission Output

part01test06.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test06.solution

```
Team 01
29 77 14 84 46 43
Team 02
15 53 59 31 55 58 21
The two teams can differ by at most 1 pound(s).
```

stderr

part01test06.err

4.1.7 Test 07

diff

part01test07.diff

Input

part01test07.input

```
99 91 34 24 74 54 34 90 11 12 52 34 77 99 69 64 42 83 63 61 79 53 67 85 77 68
```

Submission Output

part01test07.output

```
The two teams can differ by at most 0 pound(s).
```

Solution Output

part01test07.solution

```
Team 01
64 69 99 77 12 11 90 54 74 24 34 91 99
Team 02
34 52 34 42 83 63 61 79 53 67 85 77 68
The two teams can differ by at most 0 pound(s).
```

stderr

part01test07.err

4.1.8 Test 08

diff

part01test08.diff

Input

part01test08.input

```
86 30 35 47 66 72 31 42 12 72 22 87 80 47 93 50 53 59 99 86 22
```

Submission Output

part01test08.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test08.solution

```
Team 01
80 87 22 72 12 42 31 72 66 47 35 30
Team 02
86 47 93 50 53 59 99 86 22
The two teams can differ by at most 1 pound(s).
```

stderr

part01test08.err

4.1.9 Test 09

diff

part01test09.diff

Input

part01test09.input

```
77 66 62
```

Submission Output

part01test09.output

```
The two teams can differ by at most 51 pound(s).
```

Solution Output

part01test09.solution

```
Team 01
77
Team 02
66 62
The two teams can differ by at most 51 pound(s).
```

stderr

part01test09.err

4.1.10 Test 10

diff

part01test10.diff

Input

part01test10.input

```
14 34 33 38 11 42 36 31 36 57 75 56 68 80 92 12 39 20 74 93 16 47 95 56 31 40 61 18 53
```

Submission Output

part01test10.output

```
The two teams can differ by at most 0 pound(s).
```

Solution Output

part01test10.solution

```
Team 01
12 92 80 68 56 75 57 31 36 42 11 38 33 34 14
Team 02
36 39 20 74 93 16 47 95 56 31 40 61 18 53
The two teams can differ by at most 0 pound(s).
```

stderr

part01test10.err

4.1.11 Test 11

diff

part01test11.diff

Input

part01test11.input

```
27 95 27 46 92 57 57 81 59 15 72 95
```

Submission Output

part01test11.output

```
The two teams can differ by at most 1 pound(s).
```

Solution Output

part01test11.solution

```
Team 01
59 81 57 92 46 27
Team 02
95 27 57 15 72 95
The two teams can differ by at most 1 pound(s).
```

stderr

part01test11.err

4.1.12 Test 12

diff

part01test12.diff

Input

part01test12.input

```
72 69
```

Submission Output

part01test12.output

```
The two teams can differ by at most 3 pound(s).
```

Solution Output

part01test12.solution

```
Team 01
69
Team 02
72
The two teams can differ by at most 3 pound(s).
```

stderr

part01test12.err

4.1.13 Test 13

diff

part01test13.diff

Input

part01test13.input

```
53 41 43 50 81
```

Submission Output

part01test13.output

```
The two teams can differ by at most 0 pound(s).
```

Solution Output

part01test13.solution

```
Team 01
50 43 41
Team 02
53 81
The two teams can differ by at most 0 pound(s).
```

stderr

part01test13.err

4.1.14 Test 14

diff

part01test14.diff

Input

part01test14.input

```
13 18 76 78 99 89 26 90 76 14 59 54 82
```

Submission Output

part01test14.output

```
The two teams can differ by at most 0 pound(s).
```

Solution Output

part01test14.solution

```
Team 01
90 89 99 78 18 13
Team 02
76 26 76 14 59 54 82
The two teams can differ by at most 0 pound(s).
```

stderr

part01test14.err

4.1.15 Test 15

diff

part01test15.diff

Input

part01test15.input

```
45 14 39 49
```

Submission Output

part01test15.output

```
The two teams can differ by at most 21 pound(s).
```

Solution Output

part01test15.solution

```
Team 01
49 14
Team 02
45 39
The two teams can differ by at most 21 pound(s).
```

stderr

part01test15.err

4.1.16 Source Code

csce310h0mework04part01.h

```
1 #ifndef CSCE310H0MEWORK01PART01_H
2 #define CSCE310H0MEWORK01PART01_H
3 #include <vector>
4 using namespace std;
5
6 int tugOfWar( vector<int> );
7
8 #endif
```

csce310h0mework04part01.cpp

```
1 /**
2  * Author: Fateh Karan Singh Sandhu
3  * Date: 19 November 2019
4  *
5  * This program takes in a vector of weights as an input and then optimally
6  * divides them into two teams with least difference in total weights
7  *
```

```
 8 **/
 9
10 #include "csce310h0mework04part01.h"
11 #include <vector>
12
13 using namespace std;
14
15 int tugOfWar( vector<int> weight ){
16
17   int sumOfWeights = 0;
18
19   for (int i = 0 ; i < weight.size() ; i++) {
20     sumOfWeights += weight[i]; //get total weight
21   }
22
23   int sizeOfMatrix = (sumOfWeights/2); //get the size of matrix
24
25   vector< vector<int> > matrix(weight.size()+1); //create memoization vector
26   vector<int> column; //create column to push in
27   column.push_back(1);
28   for (int i = 1 ; i <= sizeOfMatrix ; i++) {
29     column.push_back(0);
30   }
31
32   for (int i = 0 ; i <= weight.size() ; i++) {
33     matrix[i] = column;
34   }
35
36   for (int i = 1 ; i <= weight.size() ; i++) {
37     for (int j = 1 ; j <= sizeOfMatrix ; j++) {
38       if (matrix[i-1][j] == 1) {
39         matrix[i][j] = 1;
40       } else {
41         if (weight[i-1] > j) {
42           matrix[i][j] = 0;
43         } else {
44           matrix[i][j] = matrix[i-1][j-weight[i-1]];
45         }
46       }
47     }
48   }
49
50   int c = 0;
51
52   for (c = sizeOfMatrix ; c != 0 ; c--) {
53     if (matrix[weight.size()][c] == 1) {
54       break;
55     }
56   }
57
58   return sumOfWeights-2*c; //return least difference
59 }
```

4.2 csce310h0mework04part02

4.2.1 Test 01

diff

part02test01.diff

Input

part02test01.a.input

```
5 6 2 7 3 4 4 4 1 4 8 3 4 3
```

part02test01.b.input

```
4 4 2 3 9 2 9 6 2 5 2 4 9 7
```

part02test01.aToB.input

```
8 3 6 4 8 9 7 8 2 1 9 5 2
```

part02test01.bToA.input

```
5 6 1 8 6 7 1 2 6 7 6 4 6
```

Submission Output

part02test01.output

```
The shortest time to complete the race is 56 seconds.
```

Solution Output

part02test01.solution

```
  58    53    47    45    38    35    31    27    23    22    18    10     7     3     0
  56    52    48    49    48    39    37    30    24    22    17    15    16     7     0
The shortest time to complete the race is 56 seconds.
```

stderr

part02test01.err

4.2.2 Test 02

diff

part02test02.diff

Input

part02test02.a.input

```
7 9 2 1 3 9 1 3 7 8 9 3 5 3 8 6 6 3 5
```

part02test02.b.input

```
2 2 1 7 9 5 6 8 8 6 7 7 2 8 3 2 8 5 2
```

part02test02.aToB.input

```
2 6 4 6 9 2 1 1 5 8 9 6 7 3 7 8 2 2
```

part02test02.bToA.input

```
2 5 4 6 8 1 7 3 1 7 6 4 1 5 1 1 9 7
```

Submission Output

part02test02.output

```
The shortest time to complete the race is 83 seconds.
```

Solution Output

part02test02.solution

```
  90    85    76    74    73    70    61    60    57    50    42    33    30    25    27    19    13
     7     5     0
  83    81    79    83    76    67    70    64    56    48    42    35    28    27    19    16    15
     7     2     0
The shortest time to complete the race is 83 seconds.
```

stderr

part02test02.err

4.2.3 Test 03

diff

part02test03.diff

Input

part02test03.a.input

```
2 2 2 3 2 9 9 1 7
```

part02test03.b.input

```
9 2 8 9 1 6 2 5 4
```

part02test03.aToB.input

```
4 6 5 3 6 1 6 5
```

part02test03.bToA.input

```
2 2 7 5 2 4 2 5
```

Submission Output

part02test03.output

```
The shortest time to complete the race is 30 seconds.
```

Solution Output

part02test03.solution

```
  30    28    26    24    23    21    17     8     7     0
  39    30    35    27    18    17    11     9     4     0
The shortest time to complete the race is 30 seconds.
```

stderr

part02test03.err

4.2.4 Test 04

diff

part02test04.diff

Input

part02test04.a.input

```
1 5 4 7 6 6 8 5 4 1 2 1 5 5 6 5 4
```

part02test04.b.input

```
8 7 9 3 7 4 7 8 5 8 4 5 5 4 9 4 2
```

part02test04.aToB.input

```
5 3 4 8 3 6 1 9 5 6 2 3 3 2 8 2
```

part02test04.bToA.input

```
7 1 4 1 3 2 5 2 9 9 4 5 9 3 2 5
```

Submission Output

part02test04.output

```
The shortest time to complete the race is 75 seconds.
```

Solution Output

part02test04.solution

```
  75   74   69   65   58   52   46   38   33   29   28   26   25   20   15    9    4
     0
  85   77   71   62   59   52   50   43   43   41   33   29   24   19   15    6    2
     0
The shortest time to complete the race is 75 seconds.
```

stderr

part02test04.err

4.2.5 Test 05

diff

part02test05.diff

Input

part02test05.a.input

```
1 3 7 3
```

part02test05.b.input

```
3 7 3 9
```

part02test05.aToB.input

```
7 3 2
```

part02test05.bToA.input

```
8 2 8
```

Submission Output

part02test05.output

```
The shortest time to complete the race is 14 seconds.
```

Solution Output

part02test05.solution

```
  14   13   10    3    0
  22   19   12    9    0
The shortest time to complete the race is 14 seconds.
```

stderr

part02test05.err

4.2.6 Test 06

diff

part02test06.diff

Input

part02test06.a.input

```
4 6 5 1 4 7 6 6 9 3 9 7
```

part02test06.b.input

```
7 1 8 3 6 1 3 6 5 7 8 4
```

part02test06.aToB.input

```
1 4 3 3 9 5 1 7 8 9 5
```

part02test06.bToA.input

```
9 3 5 4 9 1 1 6 6 3 1
```

Submission Output

part02test06.output

```
The shortest time to complete the race is 57 seconds.
```

Solution Output

part02test06.solution

```
  57    55    49    44    47    44    37    34    28    19    16     7     0
  59    52    51    43    40    34    33    30    24    19    12     4     0
The shortest time to complete the race is 57 seconds.
```

stderr

part02test06.err

4.2.7 Test 07

diff

part02test07.diff

Input

part02test07.a.input

```
1 8 6 7 2 4 5 8 3 5 4 2 5 8 8 7 7 9 5 2
```

part02test07.b.input

```
1 9 3 5 8 4 5 6 2 2 2 3 6 1 5 6 1 1 9 8
```

part02test07.aToB.input

```
9 3 7 2 1 5 6 5 3 7 1 4 8 7 7 6 3 5 6
```

part02test07.bToA.input

```
7 7 9 9 3 9 8 2 3 1 8 9 1 6 8 8 3 7 7
```

Submission Output

part02test07.output

```
The shortest time to complete the race is 83 seconds.
```

Solution Output

part02test07.solution

```
  83   82   74   68   61   63   60   55   47   47   42   40   41   42   37   29   23
    16    7    2    0
  84   83   74   71   66   58   54   49   43   41   39   37   34   28   27   22   16
    15   17    8    0
The shortest time to complete the race is 83 seconds.
```

stderr

part02test07.err

4.2.8 Test 08

diff

part02test08.diff

Input

part02test08.a.input

```
2 9 8 2 5 9 9 5 7 1 4 6 5 8 7 2 7
```

part02test08.b.input

```
5 1 5 8 3 1 6 2 3 7 7 8 6 6 6 2 2
```

part02test08.aToB.input

```
4 6 6 8 6 3 8 8 7 2 3 1 8 3 5 6
```

part02test08.bToA.input

```
2 9 4 1 4 2 9 7 7 7 5 1 4 5 9 6
```

Submission Output

part02test08.output

```
The shortest time to complete the race is 75 seconds.
```

Solution Output

part02test08.solution

```
  76   83   74   66   64   64   55   46   41   34   33   29   26   21   16    9    7
    0
  75   70   69   64   56   53   52   46   44   44   37   30   22   16   10    4    2
    0
The shortest time to complete the race is 75 seconds.
```

stderr

part02test08.err

4.2.9 Test 09

`diff`

part02test09.diff

Input

part02test09.a.input

```
7 8 6 5 8 7 4 8 5 2 9 7 4 5 8 8 1
```

part02test09.b.input

```
7 2 6 3 8 1 6 3 8 8 8 2 3 4 6 2 2
```

part02test09.aToB.input

```
4 6 1 8 2 8 3 3 3 8 8 3 5 4 9 8
```

part02test09.bToA.input

```
2 1 2 2 7 4 9 5 6 2 5 1 7 6 1 3
```

Submission Output

part02test09.output

```
The shortest time to complete the race is 79 seconds.
```

Solution Output

part02test09.solution

```
  83    79    71    68    63    60    53    50    42    37    36    27    23    19    17     9     1
     0
  79    72    70    64    61    53    52    46    43    35    27    19    17    14    10     4     2
     0
The shortest time to complete the race is 79 seconds.
```

`stderr`

part02test09.err

4.2.10 Test 10

`diff`

part02test10.diff

Input

part02test10.a.input

```
1 5 2 1 2 9 4 4 2 6 1 6 2 5 7
```

part02test10.b.input

```
8 9 9 6 9 1 7 9 4 1 6 3 2 4 7
```

part02test10.aToB.input

```
4 8 3 3 2 2 8 7 1 3 1 1 7 9
```

part02test10.bToA.input

```
4 9 1 4 8 8 3 9 4 1 8 7 6 7
```

Submission Output

part02test10.output

```
The shortest time to complete the race is 51 seconds.
```

Solution Output

part02test10.solution

```
  51    50    45    43    42    40    31    27    23    24    18    20    14    12     7     0
  62    62    53    52    47    38    37    33    24    20    22    16    13    11     7     0
The shortest time to complete the race is 51 seconds.
```

stderr

part02test10.err

4.2.11 Test 11

diff

part02test11.diff

Input

part02test11.a.input

```
9 2 9 1 4 4 3 1 6
```

part02test11.b.input

```
8 6 1 7 7 7 3 3 2
```

part02test11.aToB.input

```
8 7 3 7 5 8 1 8
```

part02test11.bToA.input

```
1 6 7 6 6 1 6 8
```

Submission Output

part02test11.output

```
The shortest time to complete the race is 38 seconds.
```

Solution Output

part02test11.solution

```
  38    29    27    18    17    13     9     7     6     0
  38    32    26    29    22    15     8     5     2     0
The shortest time to complete the race is 38 seconds.
```

stderr

part02test11.err

4.2.12 Test 12

diff

part02test12.diff

Input

part02test12.a.input

```
9 5 4 9 6 7 6 1 8 3 6 5
```

part02test12.b.input

```
6 2 8 9 2 8 4 8 9 9 9 4
```

part02test12.aToB.input

```
5 6 1 7 8 5 9 1 6 9 6
```

part02test12.bToA.input

```
5 8 3 6 1 5 7 6 4 5 5
```

Submission Output

part02test12.output

```
The shortest time to complete the race is 64 seconds.
```

Solution Output

part02test12.solution

```
  67    58    53    51    42    36    29    23    22    14    11     5     0
  64    58    56    48    39    42    34    35    27    22    13     4     0
The shortest time to complete the race is 64 seconds.
```

stderr

part02test12.err

4.2.13 Test 13

diff

part02test13.diff

Input

part02test13.a.input

```
8 3 7 6 3 6 5 4 8 8 4 5
```

part02test13.b.input

```
7 2 9 7 6 2 2 7 7 2 6 4
```

part02test13.aToB.input

```
7 3 7 2 3 4 8 7 3 5 6
```

part02test13.bToA.input

```
9 6 3 4 1 1 8 1 6 1 8
```

Submission Output

part02test13.output

```
The shortest time to complete the race is 60 seconds.
```

Solution Output

part02test13.solution

```
  60    52    49    42    36    38    32    27    23    17     9     5     0
  61    54    52    43    36    30    28    26    19    12    10     4     0
The shortest time to complete the race is 60 seconds.
```

stderr

part02test13.err

4.2.14 Test 14

diff

part02test14.diff

Input

part02test14.a.input

```
3 4 3 1 8 2 6 7 8 4 8 9 5
```

part02test14.b.input

```
7 5 2 3 8 9 9 1 3 6 1 4 7
```

part02test14.aToB.input

```
7 4 3 1 4 1 9 9 2 6 3 7
```

part02test14.bToA.input

```
2 4 2 4 3 9 8 4 1 2 5 8
```

Submission Output

part02test14.output

```
The shortest time to complete the race is 53 seconds.
```

Solution Output

part02test14.solution

```
  53    50    46    43    42    34    37    35    28    22    22    14     5     0
  59    52    47    48    45    40    31    22    21    18    12    11     7     0
The shortest time to complete the race is 53 seconds.
```

stderr

part02test14.err

4.2.15 Test 15

diff

part02test15.diff

Input

part02test15.a.input

```
5 1 3 4 6 5 7 8 4 2 3 6 9 4 2 7
```

part02test15.b.input

```
9 8 9 6 5 4 5 7 4 5 9 9 8 3 7 9
```

part02test15.aToB.input

```
4 3 5 8 1 4 4 3 5 6 2 2 8 6 7
```

part02test15.bToA.input

```
2 3 2 7 1 1 4 7 5 4 3 7 8 3 1
```

Submission Output

part02test15.output

```
The shortest time to complete the race is 76 seconds.
```

Solution Output

part02test15.solution

```
  76   71   70   67   63   57   52   45   37   33   31   28   22   13    9    7    0
  82   81   77   68   62   57   54   49   42   40   40   32   23   15   15    9    0
The shortest time to complete the race is 76 seconds.
```

stderr

part02test15.err

4.2.16 Source Code

csce310h0mework04part02.h

```
1 #ifndef CSCE310H0MEWORK04PART02_H
2 #define CSCE310H0MEWORK04PART02_H
3 #include <vector>
4 using namespace std;
5
6 int footRace( vector<int> , vector<int> , vector<int> , vector<int> );
7
8 #endif
```

csce310h0mework04part02.cpp

```
1 /**
2  * Author: Fateh Karan Singh Sandhu
3  * Date: 19 November 2019
4  *
5  * This program takes in a vector of two lane segments and time penalties
6  * and returns the shortest time to finish the race
7  *
8 **/
9
10 #include <vector>
11 #include "csce310h0mework04part02.h"
12 #include <vector>
13 #include <iostream>
14 #include <cstdio>
15
16 using namespace std;
17
18 int footRace( vector<int> laneA , vector<int> laneB , vector<int> aToB , vector<int>
       bToA ){
19
20   vector<int> row1; //create row1
21   vector<int> row2; //create row2
22
23   for (int a = 0 ; a <= laneA.size() ; a++) {
24     row1.push_back(0);
25     row2.push_back(0);
26   }
```

```
27
28    vector< vector<int> > matrix;
29    matrix.push_back(row1);
30    matrix.push_back(row2);
31
32    matrix[0][0] = 0;
33    matrix[1][0] = 0;
34
35    matrix[0][1] = laneA[laneA.size()-1];
36    matrix[1][1] = laneB[laneB.size()-1];
37
38      for (int j = 2 ; j <= laneA.size() ; j++) {
39        matrix[0][j] = min(matrix[0][j-1]+laneA[laneA.size()-j], matrix[1][j-1] + laneA[
      laneA.size()-j] + aToB[aToB.size()-j+1]);
40        matrix[1][j] = min(matrix[1][j-1]+laneB[laneB.size()-j], matrix[0][j-1] + laneB[
      laneB.size()-j] + bToA[bToA.size()-j+1]);
41      }
42
43    return min(matrix[0][laneA.size()], matrix[1][laneB.size()]); //return shortest time
44  }
```

4.3 csce310h0mework04part03

4.3.1 Test 01

diff

part03test01.diff

Input

part03test01.matrix.input

```
-1 80 -1 36 59
19 57 46 38 66
56 21 45 88 61
-1 83 58 86 -1
99 11 41 90 97
79 57 12 38 -1
-1 85 23 79 68
29 24 79 77 27
-1 60 34 76 30
86 57 93 42 -1
34 82 88 73 38
```

Submission Output

part03test01.output

```
$915 can be made.
```

Solution Output

part03test01.solution

```
    0   80    0   36   95
   19  137  183  221  287
   75  158  228  316  377
    0  241  299  402    0
   99  252  340  492  589
  178  309  352  530    0
    0  394  417  609  677
   29  418  497  686  713
```

```
    0  478  531  762  792
   86  535  628  804    0
  120  617  716  877  915
$915 can be made.
```

stderr

part03test01.err

4.3.2 Test 02

diff

part03test02.diff

Input

part03test02.matrix.input

```
53 82 19 98 42 -1 11 61
24 75 58 91 33 91 65 94
44 50 -1 89 97 77 20 -1
36 87 42 66 30 92 50 50
```

Submission Output

part03test02.output

```
$814 can be made.
```

Solution Output

part03test02.solution

```
   53  135  154  252  294    0   11   72
   77  210  268  359  392  483  548  642
  121  260    0  448  545  622  642    0
  157  347  389  514  575  714  764  814
$814 can be made.
```

stderr

part03test02.err

4.3.3 Test 03

diff

part03test03.diff

Input

part03test03.matrix.input

```
63 18 63 20 18 -1 98 98 40 84 70 92 28 -1 92 53 99
68 36 89 65 95 66 89 32 -1 63 45 52 29 82 36 62 89
68 14 36 12 61 96 79 42 -1 70 69 54 -1 35 44 29 34
17 86 52 87 -1 43 -1 19 91 33 28 82 70 44 20 41 50
42 29 36 63 55 16 56 62 58 68 -1 -1 32 38 24 49 62
-1 33 63 -1 61 40 57 55 -1 30 94 26 92 20 42 13 11
-1 24 27 95 88 -1 53 12 64 20 17 96 -1 -1 84 20 61
89 -1 99 27 19 62 27 94 97 57 86 31 59 42 46 54 27
36 96 71 26 47 23 15 54 52 16 13 45 76 64 99 74 93
34 53 -1 91 28 26 61 20 48 91 27 63 23 20 60 95 70
64 67 37 39 49 -1 73 37 31 39 69 68 31 83 80 11 -1
```

```
74 32 88 21 84 27 73 -1 22 98 19 69 47 15 35 27 48
49 33 89 71 20 42 56 98 16 20 60 76 -1 29 82 -1 92
-1 -1 50 93 16 30 94 42 45 33 81 67 57 41 26 23 19
39 42 17 82 57 27 54 80 87 91 63 73 19 54 35 53 42
68 44 23 62 40 35 29 44 79 67 22 14 55 28 16 27 88
49 84 26 -1 12 -1 38 71 -1 12 16 31 80 84 70 43 36
-1 45 76 -1 64 49 71 -1 71 -1 77 16 63 58 68 89 91
```

Submission Output

part03test03.output

```
$2246 can be made.
```

Solution Output

part03test03.solution

```
   63   81  144  164  182    0   98  196  236  320  390  482  510    0   92  145  244
  131  167  256  321  416  482  571  603    0  383  435  534  563  645  681  743  832
  199  213  292  333  477  578  657  699    0  453  522  588    0  680  725  772  866
  216  302  354  441    0  621    0  718  809  842  870  952 1022 1066 1086 1127 1177
  258  331  390  504  559  637  693  780  867  935    0    0 1054 1104 1128 1177 1239
    0  364  453    0  620  677  750  835    0  965 1059 1085 1177 1197 1239 1252 1263
    0  388  480  575  708    0  803  847  911  985 1076 1181    0    0 1323 1343 1404
   89    0  579  606  727  789  830  941 1038 1095 1181 1212 1271 1313 1369 1423 1450
  125  221  650  676  774  812  845  995 1090 1111 1194 1257 1347 1411 1510 1584 1677
  159  274    0  767  802  838  906 1015 1138 1229 1256 1320 1370 1431 1570 1679 1749
  223  341  378  806  855    0  979 1052 1169 1268 1337 1405 1436 1519 1650 1690    0
  297  373  466  827  939  966 1052    0 1191 1366 1385 1474 1521 1536 1685 1717 1765
  346  406  555  898  959 1008 1108 1206 1222 1386 1446 1550    0 1565 1767    0 1857
    0    0  605  991 1007 1038 1202 1248 1293 1419 1527 1617 1674 1715 1793 1816 1876
   39   81  622 1073 1130 1157 1256 1336 1423 1514 1590 1690 1709 1769 1828 1881 1923
  107  151  645 1135 1175 1210 1285 1380 1502 1581 1612 1704 1764 1797 1844 1908 2011
  156  240  671    0 1187    0 1323 1451    0 1593 1628 1735 1844 1928 1998 2041 2077
    0  285  747    0 1251 1300 1394    0   71    0 1705 1751 1907 1986 2066 2155 2246
$2246 can be made.
```

stderr

part03test03.err

4.3.4 Test 04

diff

part03test04.diff

Input

part03test04.matrix.input

```
78 54 62 95 -1 63 95 86 -1 -1 43 -1 96 -1 14 59 91 70
99 34 57 96 89 41 87 59 38 30 12 16 15 54 93 80 -1 12
65 86 39 -1 57 51 87 94 24 14 65 80 -1 70 52 73 66 28
```

Submission Output

part03test04.output

```
$1166 can be made.
```

Solution Output

part03test04.solution

```
   78  132  194  289    0   63  158  244    0    0   43    0   96    0   14   73  164
    234
  177  211  268  385  474  515  602  661  699  729  741  757  772  826  919  999    0
    246
  242  328  367    0  531  582  689  783  807  821  886  966    0  896  971 1072 1138
   1166
$1166 can be made.
```

stderr

part03test04.err

4.3.5 Test 05

diff

part03test05.diff

Input

part03test05.matrix.input

```
20 79 48 47 89 92 -1 26 32 12 37 95 85
52 10 84 79 -1 88 99 38 52 35 14 28 20
24 14 48 84 54 72 96 76 38 30 68 98 97
38 50 78 41 63 76 59 69 58 93 -1 53 87
85 68 19 25 66 78 33 49 97 83 32 -1 61
37 97 41 79 33 66 27 97 27 30 40 33 83
-1 -1 90 18 31 -1 57 76 15 56 85 67 33
45 96 95 56 10 -1 53 -1 83 85 29 86 13
-1 36 75 53 25 76 23 57 96 77 56 -1 51
33 50 84 92 18 49 30 88 99 26 58 67 84
96 95 -1 86 75 47 27 67 59 62 47 56 74
46 43 67 11 27 74 10 25 83 23 50 23 23
12 97 72 27 -1 -1 26 45 28 86 97 69 75
```

Submission Output

part03test05.output

```
$1815 can be made.
```

Solution Output

part03test05.solution

```
   20   99  147  194  283  375    0   26   58   70  107  202  287
   72  109  231  310    0  463  562  600  652  687  701  729  749
   96  123  279  394  448  535  658  734  772  802  870  968 1065
  134  184  357  435  511  611  717  803  861  954    0 1021 1152
  219  287  376  460  577  689  750  852  958 1041 1073    0 1213
  256  384  425  539  610  755  782  949  985 1071 1113 1146 1296
    0    0  515  557  641    0  839 1025 1040 1127 1212 1279 1329
   45  141  610  666  676    0  892    0 1123 1212 1241 1365 1378
    0  177  685  738  763  839  915  972 1219 1296 1352    0 1429
   33  227  769  861  879  928  958 1060 1318 1344 1410 1477 1561
  129  322    0  947 1022 1069 1096 1163 1377 1439 1486 1542 1635
  175  365  432  958 1049 1143 1153 1188 1460 1483 1536 1565 1658
  187  462  534  985    0    0 1179 1233 1488 1574 1671 1740 1815
$1815 can be made.
```

stderr

part03test05.err

4.3.6 Test 06

diff

part03test06.diff

Input

part03test06.matrix.input

```
-1 63 87 43 93 23 38 58 24 -1 32 52 54 25 79 99 -1 -1 66
-1 35 46 99 46 55 57 90 85 91 21 71 -1 66 65 28 41 31 85
51 62 -1 20 33 71 77 49 13 13 43 96 26 72 70 19 12 -1 86
22 32 31 23 24 95 65 39 -1 76 -1 48 43 14 48 30 90 35 52
```

Submission Output

part03test06.output

```
$1330 can be made.
```

Solution Output

part03test06.solution

```
    0   63  150  193  286  309  347  405  429    0   32   84  138  163  242  341    0
      0   66
    0   98  196  295  341  396  453  543  628  719  740  811    0  229  307  369  410
    441  526
   51  160    0  315  374  467  544  593  641  732  783  907  933 1005 1075 1094 1106
      0  612
   73  192  223  338  398  562  627  666    0  808    0  955  998 1019 1123 1153 1243
   1278 1330
$1330 can be made.
```

stderr

part03test06.err

4.3.7 Test 07

diff

part03test07.diff

Input

part03test07.matrix.input

```
93 -1 13 14 56 67 39 16 11 61 -1
86 31 23 83 71 55 74 15 88 50 62
```

Submission Output

part03test07.output

```
$731 can be made.
```

Solution Output

part03test07.solution

```
   93    0   13   27   83  150  189  205  216  277    0
  179  210  233  316  387  442  516  531  619  669  731
$731 can be made.
```

stderr

part03test07.err

4.3.8 Test 08

diff

part03test08.diff

Input

part03test08.matrix.input

```
48 93 20 91
95 37 84 85
-1 13 38 65
78 41 83 -1
42 71 53 94
66 50 45 96
72 -1 -1 77
40 40 88 -1
31 -1 53 88
```

Submission Output

part03test08.output

```
$567 can be made.
```

Solution Output

part03test08.solution

```
   48  141  161  252
  143  180  264  349
    0  193  302  414
   78  234  385    0
  120  305  438  532
  186  355  483  628
  258    0    0  705
  298  338  426    0
  329    0  479  567
$567 can be made.
```

stderr

part03test08.err

4.3.9 Test 09

diff

part03test09.diff

Input

part03test09.matrix.input

```
73 80 41 90 82 63 79 26 21 12 93 92 66 26 -1
82 41 15 55 21 34 26 63 14 32 85 77 86 62 -1
48 20 80 98 -1 69 62 64 27 42 24 37 24 46 49
13 18 21 78 55 15 76 -1 81 82 63 64 92 35 40
89 29 29 74 20 49 84 86 -1 63 63 21 39 82 22
```

```
20 89 64 66 62 63 44 27 89 93 -1 67 57 -1 41
57 96 31 96 -1 92 28 99 76 19 88 86 58 44 32
82 60 48 46 96 17 87 78 22 19 95 25 80 90 82
-1 93 78 72 12 -1 12 98 55 -1 85 79 82 58 90
38 13 73 35 -1 66 89 -1 -1 84 82 46 63 85 87
81 -1 54 43 34 90 45 -1 39 98 29 46 -1 98 70
11 44 -1 95 17 10 96 18 -1 71 22 -1 -1 79 81
31 16 45 17 91 91 40 50 -1 -1 48 44 68 36 11
72 52 71 80 95 65 85 64 55 73 68 86 -1 41 54
10 -1 -1 86 56 29 28 31 52 -1 11 22 91 10 91
```

Submission Output

part03test09.output

```
$2085 can be made.
```

Solution Output

part03test09.solution

```
   73   153   194   284   366   429   508   534   555   567   660   752   818   844     0
  155   196   211   339   387   463   534   597   611   643   745   829   915   977     0
  203   223   303   437     0   532   596   661   688   730   769   866   939  1023  1072
  216   241   324   515   570   585   672     0   769   851   914   978  1070  1105  1145
  305   334   363   589   609   658   756   842     0   914   977   999  1109  1191  1213
  325   423   487   655   717   780   824   869   958  1051     0  1066  1166     0  1254
  382   519   550   751     0   872   900   999  1075  1094  1182  1268  1326  1370  1402
  464   579   627   797   893   910   997  1077  1099  1118  1277  1302  1406  1496  1578
    0   672   750   869   905     0  1009  1175  1230     0  1362  1441  1523  1581  1671
   38   685   823   904     0    66  1098     0     0    84  1444  1490  1586  1671  1758
  119     0   877   947   981  1071  1143     0    39   182  1473  1536     0  1769  1839
  130   174     0  1042  1059  1081  1239  1257     0   253  1495     0     0  1848  1929
  161   190   235  1059  1150  1241  1281  1331     0     0  1543  1587  1655  1884  1940
  233   285   356  1139  1245  1310  1395  1459  1514  1587  1655  1741     0  1925  1994
  243     0     0  1225  1301  1339  1423  1490  1566     0  1666  1763  1854  1935  2085
$2085 can be made.
```

stderr

part03test09.err

4.3.10 Test 10

diff

part03test10.diff

Input

part03test10.matrix.input

```
43 13 44 50 -1 -1 -1 99 90 12 23
-1 84 21 84 13 48 -1 38 -1 14 63
91 17 79 83 61 40 32 70 -1 92 65
33 86 53 83 84 31 59 29 53 37 68
-1 34 93 62 39 14 -1 13 73 33 52
55 58 88 71 17 53 94 44 66 81 48
71 99 42 28 18 53 25 64 97 30 35
15 95 42 25 -1 38 96 96 33 12 58
58 34 87 20 77 -1 -1 79 -1 39 50
31 59 46 71 20 -1 35 69 25 19 84
95 75 88 21 67 92 99 28 44 79 30
```

```
31 68 78 76 66 83 48 85 74 78 67
84 22 91 14 15 25 91 32 97 28 78
48 -1 40 74 30 64 31 61 34 86 11
63 90 55 14 21 29 22 39 66 90 81
-1 82 51 98 23 87 -1 67 23 82 41
-1 63 16 65 66 37 79 16 35 32 70
38 -1 53 33 40 61 56 70 62 53 81
```

Submission Output

part03test10.output

```
$1883 can be made.
```

Solution Output

part03test10.solution

```
   43    56   100   150     0     0     0    99   189   201   224
    0   140   161   245   258   306     0   137     0   215   287
   91   157   240   328   389   429   461   531     0   307   372
  124   243   296   411   495   526   585   614   667   704   772
    0   277   389   473   534   548     0   627   740   773   825
   55   335   477   548   565   618   712   756   822   903   951
  126   434   519   576   594   671   737   820   919   949   986
  141   529   571   601     0   709   833   929   962   974  1044
  199   563   658   678   755     0     0  1008     0  1013  1094
  230   622   704   775   795     0    35  1077  1102  1121  1205
  325   697   792   813   880   972  1071  1105  1149  1228  1258
  356   765   870   946  1012  1095  1143  1228  1302  1380  1447
  440   787   961   975  1027  1120  1234  1266  1399  1427  1525
  488     0  1001  1075  1105  1184  1265  1327  1433  1519  1536
  551   641  1056  1089  1126  1213  1287  1366  1499  1609  1690
    0   723  1107  1205  1228  1315     0  1433  1522  1691  1732
    0   786  1123  1270  1336  1373  1452  1468  1557  1723  1802
   38     0  1176  1303  1376  1437  1508  1578  1640  1776  1883
$1883 can be made.
```

stderr

part03test10.err

4.3.11 Test 11

diff

part03test11.diff

Input

part03test11.matrix.input

```
16 21 24
18 71 -1
-1 88 52
48 35 11
32 67 62
43 21 69
12 68 79
10 26 20
-1 -1 49
```

Submission Output

part03test11.output

```
$577 can be made.
```

Solution Output

part03test11.solution

```
   16   37   61
   34  108    0
    0  196  248
   48  231  259
   80  298  360
  123  319  429
  135  387  508
  145  413  528
    0    0  577
$577 can be made.
```

stderr

part03test11.err

4.3.12 Test 12

diff

part03test12.diff

Input

part03test12.matrix.input

```
16 -1 39 38 37 -1 90
38 30 98 39 24 19 50
```

Submission Output

part03test12.output

```
$314 can be made.
```

Solution Output

part03test12.solution

```
   16    0   39   77  114    0   90
   54   84  182  221  245  264  314
$314 can be made.
```

stderr

part03test12.err

4.3.13 Test 13

diff

part03test13.diff

Input

part03test13.matrix.input

```
47 29 80 61 -1 91 54 16 73
33 20 34 36 39 71 -1 75 -1
56 40 46 44 85 46 39 13 17
62 66 30 55 90 69 29 54 98
75 78 44 64 22 65 36 46 34
62 89 10 33 19 19 49 56 34
-1 -1 -1 39 85 83 78 -1 33
18 17 20 55 95 87 15 -1 -1
56 26 23 74 74 65 36 96 79
57 61 42 13 82 42 61 64 42
43 69 94 86 24 61 10 31 66
```

Submission Output

part03test13.output

```
$1182 can be made.
```

Solution Output

part03test13.solution

```
   47   76  156  217    0   91  145  161  234
   80  100  190  253  292  363    0  236    0
  136  176  236  297  382  428  467  480  497
  198  264  294  352  472  541  570  624  722
  273  351  395  459  494  606  642  688  756
  335  440  450  492  513  625  691  747  790
    0    0    0  531  616  708  786    0  823
   18   35   55  586  711  798  813    0    0
   74  100  123  660  785  863  899  995 1074
  131  192  234  673  867  909  970 1059 1116
  174  261  355  759  891  970  980 1090 1182
$1182 can be made.
```

stderr

part03test13.err

4.3.14 Test 14

diff

part03test14.diff

Input

part03test14.matrix.input

```
66 42 77 73 42 71 -1 41 60 79 50 -1 90 52 -1 93
55 90 62 40 48 25 63 -1 95 61 -1 17 65 40 63 11
37 99 63 82 90 19 93 12 80 -1 89 55 20 15 58 12
16 68 -1 28 99 69 -1 87 15 44 73 54 85 35 43 15
77 87 65 -1 50 51 16 86 25 -1 -1 -1 73 41 33 76
43 50 78 87 57 39 70 75 64 41 22 97 95 71 24 72
84 19 30 56 36 40 49 44 97 32 11 52 73 72 54 96
93 32 37 25 57 73 25 22 40 57 41 25 38 24 26 70
76 -1 54 47 69 24 41 53 52 91 22 -1 94 78 20 90
39 42 60 36 -1 25 39 40 34 58 88 96 12 65 80 55
98 52 73 79 89 49 -1 81 63 11 90 94 52 52 88 58
20 38 10 33 18 17 65 31 87 72 66 45 95 97 72 13
```

Submission Output

part03test14.output

```
$1962 can be made.
```

Solution Output

part03test14.solution

```
   66  108  185  258  300  371    0   41  101  180  230    0   90  142    0   93
  121  211  273  313  361  396  459    0  196  257    0   17  155  195  258  269
  158  310  373  455  545  564  657  669  749    0   89  144  175  210  316  328
  174  378    0  483  644  713    0  756  771  815  888  942 1027 1062 1105 1120
  251  465  530    0  694  764  780  866  891    0    0    0 1100 1141 1174 1250
  294  515  608  695  752  803  873  948 1012 1053 1075 1172 1267 1338 1362 1434
  378  534  638  751  788  843  922  992 1109 1141 1152 1224 1340 1412 1466 1562
  471  566  675  776  845  918  947 1014 1149 1206 1247 1272 1378 1436 1492 1632
  547    0  729  823  914  942  988 1067 1201 1297 1319    0 1472 1550 1570 1722
  586  628  789  859    0  967 1027 1107 1235 1355 1443 1539 1551 1616 1696 1777
  684  736  862  941 1030 1079    0 1188 1298 1366 1533 1633 1685 1737 1825 1883
  704  774  872  974 1048 1096 1161 1219 1385 1457 1599 1678 1780 1877 1949 1962
$1962 can be made.
```

stderr

part03test14.err

4.3.15 Test 15

diff

part03test15.diff

Input

part03test15.matrix.input

```
23 89 47 98 77 -1 50 81
30 19 34 27 92 77 87 99
```

Submission Output

part03test15.output

```
$689 can be made.
```

Solution Output

part03test15.solution

```
   23  112  159  257  334    0   50  131
   53  131  193  284  426  503  590  689
$689 can be made.
```

stderr

part03test15.err

4.3.16 Source Code

csce310h0mework04part03.h

```
1 #ifndef CSCE310H0MEWORK01PART03_H
2 #define CSCE310H0MEWORK01PART03_H
3 #include <vector>
4 using namespace std;
5
6 int collectWithLoss( vector< vector<int> > );
7
8 #endif
```

```cpp
/**
 * Author: Fateh Karan Singh Sandhu
 * Date: 19 November 2019
 *
 * This program takes in a grid  as an input and returns the most
 * money that can be made
 *
**/

#include "csce310h0mework04part03.h"
#include <vector>

using namespace std;

int collectWithLoss( vector< vector<int> > grid ){

  //set all -1's to 0's
  for (int i = 0 ; i < grid.size() ; i++) {
    for (int j = 0 ; j < grid[i].size() ; j++) {
      if (grid[i][j] == -1) {
        grid[i][j] = 0;
      }
    }
  }

  for (int i = 1 ; i < grid.size() ; i++) {
   if(grid[i][0] != 0) {
    grid[i][0] = grid[i-1][0] + grid[i][0];
   }
  }

  for (int i = 1 ; i < grid[0].size() ; i++) {
   if(grid[0][i] != 0) {
    grid[0][i] = grid[0][i-1] + grid[0][i];
   }
  }

  for (int i = 1 ; i < grid.size() ; i++) {
    for (int j = 1 ; j < grid[i].size() ; j++) {
      if (grid[i][j] != 0) {
        grid[i][j] = max(grid[i-1][j]+grid[i][j], grid[i][j-1] + grid[i][j]);
      }
    }
  }

  return grid[grid.size()-1][grid[0].size()-1];
}
```