CS 218 – MIPS Assignment #4

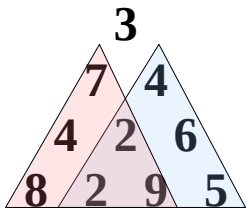
Purpose: Become familiar with the MIPS Instruction Set, and the MIPS function standard calling convention, and indexing for multiple dimension arrays.
Due: Wednesday 4/19
Points: 120

Assignment

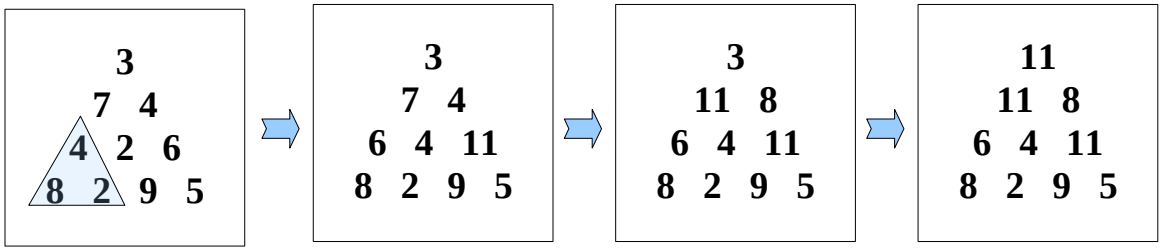
Given a triangle of numbers, with an order of 4, as shown (on right), we wish to find a path from the top (3 here) to the bottom row with the *least* cost. At each step, left or right is the only option. The 'cost' is a summation of the numbers used along the way. For example, taking the outside path on the right (3, 4, 6, 5) would yield a sum of 18. Taking the path (3, 7, 2, 2) would yield a sum of 14. However, taking the path (3, 4, 2, 2) would yield a sum of 11, which is the least cost path for this triangle.

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

There is an efficient algorithm to solve this problem using a dynamic programming[1] approach. Dynamic programming is a method for solving a problem by breaking it down into a collection of simpler sub-problems, solving each of those sub-problems, and storing their solutions.



Standing at the top of the triangle we have to choose between going left and right. In order to make the optimal choice (which minimizes the overall sum), we would have to know the sum we can get if we go either way. So in order to answer the question we would basically have to solve the two smaller problems. We can break each of the sub-problems down in a similar way, and we can continue to do so until we reach a sub-problem at the bottom line consisting of only one number, then the question becomes trivial to answer, since the answer is the number it self. Once that question is answered we can move up one line, and answer the questions posed there with a solution which is ***number*** + *min*(***left***,***right***) for each entry in that row. For example, in the third row, first number (4), we would use 4+2 which is smaller than 4+8 and over-write the 4 with a 6. Once we know the answer to all 3 sub-problems on the next to last line, we can move up and answer the posed sub-problems by the same formula already applied. And we can continue to do so until we reach the original question of whether to go left or right. This process is shown as follows:



At this point, the minimum cost path is located at the top (11 in this example).

1 For more information, refer to: https://en.wikipedia.org/wiki/Dynamic_programming

Write a simple MIPS assembly language program to find the cost of the least cost path in a triangle of numbers. The provided main calls the following functions:

- Write a MIPS void function, ***displayTriangle()***, that will display a formatted triangle grid with the given title. Refer to the sample output for examples.
- Write a MIPS void function, ***findLeastCost()***, to find the least cost for a path from the top to the bottom of a given triangle. The function should display the final least cost and the possible paths (from top to bottom). The number of possible paths is $\mathbf{2}^{order-1}$ which is 8 in the previous example.

Array Implementation

In assembly, multi-dimension arrays are implemented as a large single dimension array. The formula for calculating two-dimensional array indexing is:

```
addr[r][c] = baseAddress + (r * colSize + c) * dataSize
```

You must use the formula to access matrix elements. No score will be provided for submissions that do not use this formula. Refer to the text, *MIPS Assembly Language Programming using QtSpim*, Chapter 10, Multi-Dimension Array Implementation for more information.

Submission

When complete, submit:

- A copy of the **source file** via the class web page before class time.

Example Output

The following is the example of the output for MIPS assignment #4. *Note*, the output is truncated.

```
MIPS Assignment #4
Program to Find Cost of Least Cost Path.

****************************************
Triangle #1 (example)


          03
        07  04
      04  02  06
    08  02  09  05

Least Cost Traversal: 11
Possible Solutions: 8

****************************************
Triangle #2 (small)


           05
         06  08
       03  04  07
     05  08  02  06
   08  05  01  03  09

Least Cost Traversal: 18
Possible Solutions: 16
```

```
****************************************
Triangle #3 (small)


             03
           07  04
         04  02  06
       08  02  09  05
     09  03  06  02  07
   01  02  08  04  07  03

Least Cost Traversal: 16
Possible Solutions: 32

****************************************




    . . .  output truncated  . . .
```

Submission

- All source files must assemble and execute with QtSpim/SPIM MIPS simulator.
- Submit source file
  - Submit a copy of the program source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time (at a maximum rate of 5 submissions per hour).
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given assignment. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
#  Name: <your name>
#  NSHE ID: <your id>
#  Section: <section>
#  Assignment: <assignment number>
#  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a reduction of points.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	3%	Must include header block in the required format (see above).
General Comments	7%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	90%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.