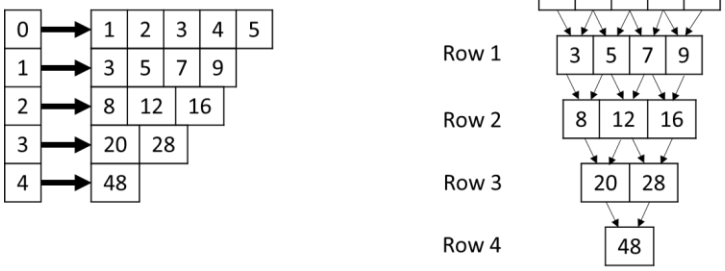
1) (10 pts) DSN (Dynamic Memory Management in C)

Starting with a 0-indexed dynamic integer 1D array called `base`, compute the triangular sum with only the **EXACT** proper amount of allocated space needed (no more or no less) by completing the user defined function definition. The triangular sum is the value of the elements present in the current dynamic array (based on the `row`) after the following process. For each `row`, the current index `i` will result in the value sum of the previous `row` (`row - 1`) at index `i` and `i + 1`. If the 2D array is named `trisum`, then the process to populate the values properly of the triangular sum will be as follows:

`trisum[row][i] = trisum[row - 1][i] + trisum[row - 1][i + 1]`

The below picture shows a nice visual representation of the triangular sum. Note that `base` is row 0.



This function will return an address to an array of arrays (dynamic 2D array) that visually represents the triangular sum. The second parameter `n` represents the number of elements in the `base` array `row` 0.

```
int ** triangularSum(int * base, int n) {

    int ** trisum = malloc(sizeof(int *) * n); //2pts
    trisum[0] = malloc(sizeof(int) * n); //1pt
    for (int i=0; i<n; i++)
        trisum[0][i] = base[i];          // 1 pt for assignment.

    for(int row = 1; row < n; ++row) { //1pt

        trisum[row] = malloc(sizeof(int) * (n - row)); //2pts

        for(int i = 0; i < n - row; ++i) //1pt
            trisum[row][i] = trisum[row-1][i] + trisum[row-1][i+1]; //1pt

     }

     return trisum; //1pt
}
```

Grading Notes: Take an integer number of points. For two small errors that you believe are each worth less than a point, take off 1 pt total. If there's only one tiny error (say arrow instead of one dot) correct it and give full credit. There are other ways to structure this.