University of Nevada, Las Vegas Computer Science 477/677 Fall 2021

Assignment 8: Due Monday November 29, 2021 11:59 pm

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. Turn in the completed assignment on canvas, using instructions given to you by the grader, Mr. Heerdt, by 11:59 PM November 1.

1. **A routing problem.** You are given an N×N rectangular grid of cells. Some cells are empty, others are blocked. The problem is to find a shortest path from the upper left cell to the lower right cell that does not pass through any blocked cell. The path can join any two empty cells that are adjacent either horizontally or vertically, but not diagonally. Write pseudocode for an algorithm that finds the shortest possible length of such a path.

   This is not a programming assignment. I only want to see pseudocode, which can be written entirely in English if you wish.

   Assume that the grid is given to you as an N×N array of type Boolean, whose value is **true** if the cell is empty and **false** if the cell is blocked.

   Here is an example of such a grid. Cells marked X are blocked; the others are empty. My solution is indicated in the adjacent matrix.

```
----------------------
|   X                |
|X          X      X |
|        X X    X    |
|X X XX  X     X     |
| X X  X  XXXX XX    |
|     X XX   XX     X|
|   X    X       X X |
|    XX  XX     XX X  |
|       X  XXX       |
|     X      X X  X  |
| X  X X XX   XX     |
|      X        X X X|
|        XXXX   XX   |
|  XX      XX   X    |
|            X    XXX|
|     X   X X   X XXX|
|XXX   X  X XX X XX  |
|X           XX X X X|
|    X X    X    X   |
|   X  XX     XX   X |
----------------------
```

```
----------------------
|oo X     oooo       |
|Xooooooooo Xooooo X |
|        X X    Xo   |
|X X XX  X     X o   |
| X X  X  XXXX XXo   |
|     X XX   XX oo  X|
|   X    X    oooX X |
|    XX  XX   oXX X  |
|   ooooX  XXXooo    |
|   o XooooooX Xo X  |
| X oX X XX ooXXoooo |
|   o  X     oooX XoX|
|   oo   XXXX  oXX o |
|  XXooo   XX  oXooo |
|      o     X oooXXX|
|     Xoo X X   X XXX|
|XXX   Xo X XX X XX  |
|X      o    XX X X X|
|    X Xoo  XooooXooo|
|   X  XXoooooXXoooXo|
----------------------
```

2. You wish to compute all entries of Pascal's triangle down to the 10th row, namely $C(n,k) = \binom{n}{k}$ using dynamic program and the recurrence $C(n,k) = C(n-1,k-1) + C(n-1,k)$ for $0 < k < n$, and $C(n,0) = C(n,n)$ for all n. To save space, you want to store Pascal's triangle as a triangular array. $C(n,k)$ will be stored as `X[location]`, where `location` is a function of n and k. Here is your code, with one line deleted: You wish to compute all entries of Pascal's triangle down to the 10th row, namely $C(n,k) = \binom{n}{k}$ using dynamic program and the recurrence $C(n,k) = C(n-1,k-1) + C(n-1,k)$ for $0 < k < n$, and $C(n,0) = C(n,n)$ for all n. To save space, you want to store Pascal's triangle as a triangular array. $C(n,k)$ will be stored as `X[location]`, where `location` is a function of n and k. Here is your code, with one line incomplete:

```
int const N = 10;
int X[N*(N+1)/2]; // The size of X is N+1 choose 2, which is 55 if N = 10

 int location(int n,int k)
  {
   assert(0 <= k and k <= n);
   return // INSERT CORRECT FORMULA HERE.
  }

  void store(int value, int n, int k)
   {
    X[location(n,k)] = value;
   }

  int fetch(int n, int k)
   {
    return X[location(n,k)];
   }

 int main()
  {
   for(int n = 0; n <= N; n++)
    for(int k = 0; k <= n; k++)
     if(k == 0 or k == n) store(1,n,k);
     else store(fetch(n-1,k-1)+fetch(n-1,k),n,k);
   // Write out the triangle
   for(int n = 0; n <= N; n++)
    {
     for(int k = 0; k <= n; k++)
      cout << " " << fetch(n,k);
     cout << endl;
    }
   return 1;
  }
```

And here is the output when I ran the program:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```