You, along with your pet dog 'Snoopy Zoo', are part of a Mystery Solving Group and have been informed that a shape-shifting ghoul has been terrorizing a local neighborhood. You plan a stealth mission to go house by house and check for the monster. However, the ghoul is smart enough to find it suspicious if you go to any two adjacent houses, causing the ghoul to get scared and change neighborhoods. Being a smart student who has taken CS/ECE 374 you decide to inspect as many people as you can in one direction in the interest of time.

1. Your job is to calculate what is the best path you can take to maximize your chances of catching the shape-shifter.

   Suppose you are given $Residents[1....n]$ as the number of residents in a house of a linear neighborhood where $Residents[i]$ is the number of residents in the ith House from the start of the neighborhood.

   For example, consider an instance where $n = 4$, $Residents = [2,4,6,2]$. Inspecting the first and the third house will allow you to check a total of $2+6 = 8$ in the neighborhood. On the other hand, if one skips the first house and inspects the second house and the fourth house, the total number of people inspected is only $4+2 = 6$.

   Describe and analyze an algorithm to determine the maximum total number of people you can inspect without alerting the ghoul given the array $Residents[1..n]$ as the input.

Solution: To simplify boundary cases, we add a sentinel value $Residents[0] = 0$.

Let $MaxPeople(i)$ denote the maximum number of people that you can inspect until the ith house from the start.

This means we will either inspect the ith house making it add the maximum search total up till $Residents[i-2]$ to total inspections or not inspecting it, resulting in a maximum inspection of up till $Residents[i-1]$.

For our result, we need to compute $MaxPeople(n)$. This function satisfies the following recurrence:

$$MaxPeople(i) = \begin{cases} 0 & \text{if } i \le 0 \\ \max\left\{\begin{array}{c} MaxPeople(i-1) \\ Residents[i] + MaxPeople(i-2) \end{array}\right\} & \text{otherwise} \end{cases}$$

We can memoize this function into an a one-dimensional array $MaxPeople[1..n]$, which we can fill from left to right.

```
MaxPeople(Residents[1..n]):
  MaxPeople[0] ← 0                    ⟨⟨sentinel⟩⟩
  MaxPeople[1] ← Residents[1]         ⟨⟨base case⟩⟩
  for i ← 2 up to n
      MaxPeople[i] ← max{MaxPeople[i−1], Residents[i] + MaxPeople[i−2]}
  return MaxAir[n]
```

Because of the linear traversal and dynamic memoization for *MaxPeople*, the algorithm runs in *O(n) time*. ■

2. Even though you cheked the most number of people you possibly could, the ghoul seems to have ran away. You follow the trails and find out that it has moved to another neighborhood. This time the ghoul decides to move into a circular neighborhood. Making your search more difficult.

   Describe and analyze an algorithm to determine the maximum total number of people you can inspect without alerting the ghoul given the array $Residents[1..n]$ as the input, this time depicting a circular neighborhood.

Solution: As in the previous problem, add a sentinel value $Residents[0] = 0$.

Now let $MaxPeople(i)$ denote the maximum number of people you can inspect up till the ith house. We need to compute $MaxPeople(n)$. This function obeys the following recurrence:

$$MaxPeople(i) = \begin{cases} 0 & \text{if } i \leq 0 \\ \max\left\{\begin{array}{c} MaxPeople(i-1) \\ Residents[i] + MaxPeople(i-2) \end{array}\right\} & \text{otherwise} \end{cases}$$

This recurrence is exactly the same as the previous problem but we will change how we use it for our needs. We essentially split the problem into two parts, with one part forced to start from the first house and end beofre reaching the last, while the other part forced to ignore the first house and ending on the last house.

Comparing the results from these two parts to find the max would provide us the value we need.

We can memoize this function into two one-dimensional arrays $VisitingFirst[1..n]$ and $VisitingLast[1..n]$, which we can fill by considering rows from left to right in a similar fashion to the previous problem.

```
MaxPeople(Residents[1..n]):
  VisitingFirst[0] ← 0                    ⟨⟨sentinel⟩⟩
  VisitingFirst[1] ← Residents[1]         ⟨⟨base case⟩⟩
  VisitingLast[0] ← 0                     ⟨⟨sentinel⟩⟩
  VisitingLast[1] ← 0                     ⟨⟨base case⟩⟩
  for i ← 2 up to n
      VisitingLast[i] ← max{VisitingLast[i−1], Residents[i] + VisitingLast[i−2]}
      if i = n
          VisitingFirst[i] ← VisitingFirst[i−1]
      else
          VisitingFirst[i] ← max{VisitingFirst[i−1], Residents[i] + VisitingFirst[i−2]}
  return max{VisitingFirst[n], VisitingLast[n]}
```

Because of the linear traversal and dynamic memoization for $MaxPeople$, the algorithm runs in $\boldsymbol{O(n)}$ ***time***. ■

3. **To think about later:** Due to *extereme bad luck*, the ghoul somehow manged to slip yet again. This time moving into a block style 2D neighborhood represented by $Residents[1..n][1..m]$ where $Residents[i][j]$ is the number of residents in the jth house on the ith lane. Now allowing you to move in two directions instead of one.

   Describe and analyze an algorithm to determine the maximum total number of people you can inspect without alerting the ghoul given the array $Residents[1..n][1..m]$ as the input, this time depicting a 2D neighborhood.

Solution: To simplify boundary cases, we add sentinel values $Residents[i][0] = 0$ and $Residents[0][j] = 0$ for all i and j.

Let $MaxPeople(i, j)$ denote the maximum number of people that you can inspect until the ith lane and jth house from the start coming from either direction.

This means we will either inspect the jth house in the ith lane making it add the maximum search total up till $Residents[i-2][j]$ or up till $Residents[i][j-2]$ to total inspections since we can now come from either direction or not inspecting it, resulting in a maximum inspection of up till $Residents[i-1][j]$ or up till $Residents[i][j-1]$.

To get our desired result we need to compute $MaxPeople(n, m)$.

This function satisfies the following recurrence:

$$MaxPeople(i,j) = \begin{cases} 0 & \text{if } i \le 0 \text{ or } j \le 0 \\ \max \begin{Bmatrix} MaxPeople(i-1,j) \\ Residents[i][j] + MaxPeople(i-2,j) \\ MaxPeople(i,j-1) \\ Residents[i][j] + MaxPeople(i,j-2) \end{Bmatrix} & \text{otherwise} \end{cases}$$

We can memoize this function into a two-dimensional array $MaxPeople[1..n, 0..m]$, which we can fill by considering rows from top to bottom in the outer loop and filling each row left to right inner loop.

```
MaxPeople(Residents[1..n][1..m]):
  for i ← 1 up to n
      MaxPeople[i][0] ← 0          ⟨⟨sentinel⟩⟩
  for j ← 1 up to m
      MaxPeople[0][j] ← 0          ⟨⟨sentinel⟩⟩

  for i ← 1 up to n
      for j ← 1 up to m
          MaxPeople[i][j] ← max { MaxPeople[i-1][j],
                                  Residents[i][j] + MaxPeople[i-2][j],
                                  MaxPeople[i][j-1],
                                  Residents[i][j] + MaxPeople[i][j-2] }
  return MaxPeople[n][n]
```

Because of the 2 dimensional memoization of $MaxPeople$, the algorithm runs in $\boldsymbol{O(n * m)}$ ***time***. ■