**Problem: House Robber II**

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle, meaning the first house is the neighbor of the last one. Adjacent houses have a security system connected, and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array nums representing the amount of money in each house, print the maximum amount of money you can rob tonight without alerting the police.

**Input Format**

* The first line contains an integer N (size of array).
* The second line contains N space-separated integers describing the array.

**Constraints**

* 1 <= N <= 100
* 0 <= nums[i] <= 1000

**Output Format**

* Print the maximum amount of money you can rob tonight without alerting the police.

**Sample Input**

3

2 3 2

**Sample Output**

3

**Explanation**

You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent houses. The maximum amount of money you can rob is from the second house (money = 3).

EXTRA TEST CASES:

TEST CASE 1:

INPUT:

100

10 9 2 1 2 2 6 1 3 5 2 6 5 9 10 1 10 2 5 5 5 1 10 10 4 4 7 5 4 10 9 1 4 3 7 1 8 3 6 7 5 9 1 3 3 3 7 5 5 6 1 6 10 2 5 9 5 2 9 7 3 10 6 6 3 3 4 3 2 6 6 5 1 2 9 1 4 4 6 6 4 2 6 1 3 10 7 4 10 4 4 1 5 9 3 9 2 4 9 10

OUTPUT:

291

TEST CASE 2:

INPUT:

100

128 172 47 18 41 86 61 194 51 34 172 106 104 26 71 109 95 80 156 137 87 163 3 13 64 192 104 114 105 27 67 24 177 195 16 124 77 118 155 33 59 44 75 142 195 52 40 140 187 83 62 40 106 113 129 98 156 184 159 81 182 200 131 175 97 124 92 169 7 114 186 39 193 180 186 151 152 122 11 84 170 29 103 185 64 97 154 62 110 88 171 83 17 154 89 49 152 182 194 89

OUTPUT:

5949

TEST CASE 3:

INPUT:

35

8 10 1 5 1 1 8 10 8 2 3 8 5 3 4 10 6 7 1 10 4 10 8 2 4 1 5 7 4 8 1 7 1 8 9

OUTPUT:

114

TEST CASE 4:

INPUT:

5

200 3 140 20 10

OUTPUT:

340

TEST CASE 5:

INPUT:

1

0

OUTPUT:

0

**Solution**

To solve this problem, we need to consider the circular nature of the houses. This means that if we rob the first house, we cannot rob the last house and vice versa. Therefore, we have two scenarios to consider:

1. Rob houses from the second house to the last house.
2. Rob houses from the first house to the second-to-last house.

We can use dynamic programming to solve both scenarios and take the maximum of the two results.

Let's implement this solution in Python:

python

Copy code

def rob(nums):

def rob\_linear(houses):

prev, curr = 0, 0

for money in houses:

prev, curr = curr, max(curr, prev + money)

return curr

n = len(nums)

if n == 1:

return nums[0]

return max(rob\_linear(nums[1:]), rob\_linear(nums[:-1]))

# Input

n = int(input().strip())

nums = list(map(int, input().strip().split()))

# Output

print(rob(nums))

This code defines a helper function rob\_linear to handle the linear (non-circular) version of the problem, and then uses it to compute the maximum money for the two possible scenarios. Finally, it prints the maximum of these two results