Assignment Week 4 - Questions and Solutions

Course: Building Blocks of Programming

Topic: Flowcharts - Arrays

Q 1. Draw a flowchart that takes as input an integer N, the size of an array, and then N integer elements of an array and print "yes" if the array is a palindrome, otherwise print "no"

Ans. We can check for each valid index i (for 0 based arrays this is from 0 to $N-1$), whether i^{th} and $(n-i-1)^{th}$ index are same or not. If and only if the condition is satisfied for all indices, the array is a palindrome. The flowchart is in Fig. 1

Grading. If the flowchart is correct then 2, if there are any errors in the flowchart but the logic is correct then 1, otherwise 0.

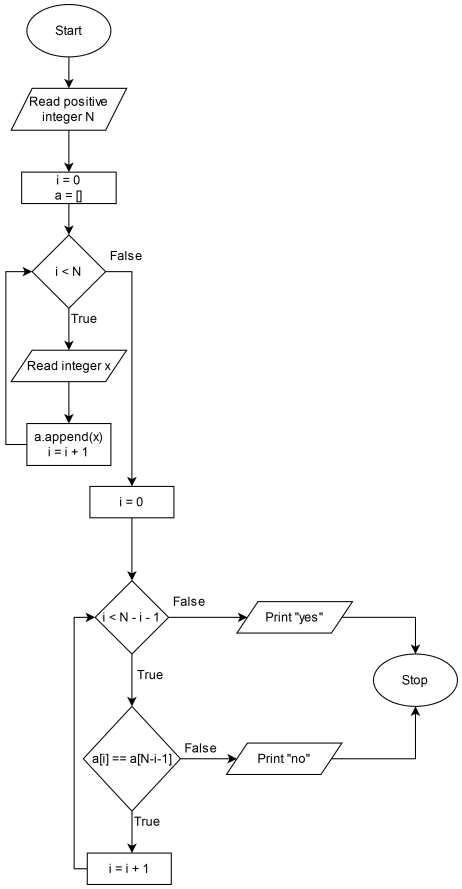

Figure 1: Sample flowchart for Q1

Q 2. Draw a flowchart that takes integers as input, until the value "0" is given, and prints all the integers that were given before "0" in reverse order. For example for the input, 1 2 3 4 9 0 The output should be 9 4 3 2 1

Ans. We should use an array to store the values so that we can revisit them in the reverse order and print them. The flowchart is in Fig 2.

Grading. If the output is correct then 2, if there are any errors in the flowchart then 1, otherwise 0.

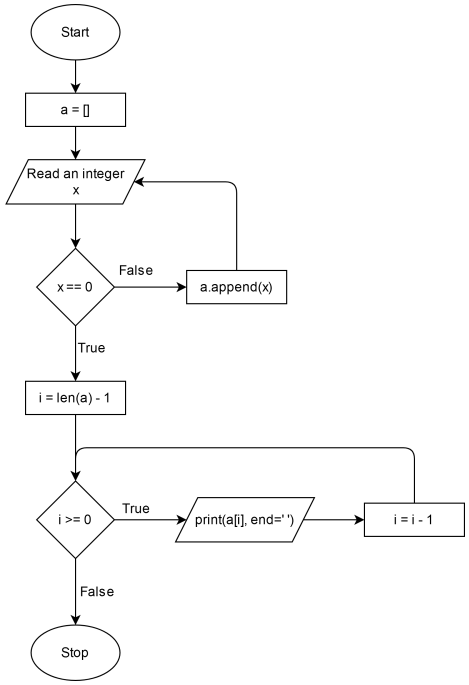

Figure 2: Sample flowchart for Q2

Q 3. Draw a flowchart that takes as input an array of size N and prints the number of occurrences of the maximum value in the array. Specifically, the inputs should be

1. a positive integer N
2. N integers denoting the elements of the array

and the output should be an integer that denotes the number of occurrences of the maximum value in the array. For example for the input, 6 3 1 3 3 2 3 the output should be, 4 because there are 4 values equal to 3.

Ans. An easy solution is to store all the elements in an array, find the maximum value and traverse the array a second time and find the count of the maximum value. This is shown in Fig 3. We can also solve this problem by just reading the elements and maintaining the maximum value of elements read so far and their count. We will update the maximum value if the next element is strictly greater than the current maximum and reset count to 1. If the next element is equal to the current maximum then increase count by 1. This optimisation is not required.

Grading. Any solution that prints the correct output for all valid inputs give 2, if there are any mistakes in the flowchart then 1, otherwise 0.

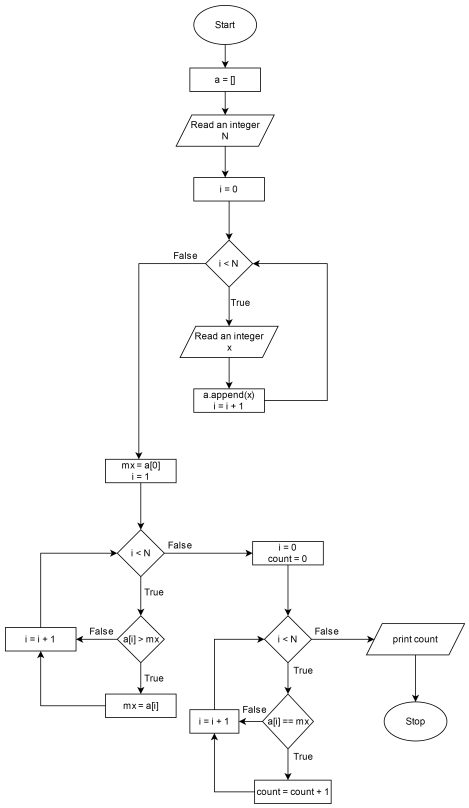

Figure 3: Sample flowchart for Q3

Q 4. Draw a flowchart that takes as input

1. a positive integer N
2. N integers

and prints all the numbers in even position (numbered from 1) first, followed by all the numbers in an odd position. For example for the input, 5 2 3 4 2 3 The output should be, 3 2 2 4 3 The numbers at even positions are "3, 2" and those at odd positions are "2, 4, 3"

Ans. We need to store all the inputs in an array so that we can traverse them in the given order. We iterate through the array twice, in the first time we print only those values such that the 0-based index is odd, in the second time we print only those values which are even. This is shown in Fig 4

Grading. Any flowchart that prints the correct output given the input give 2, if there are any errors in the flowchart but the logic is correctly identified then 1, otherwise 0.

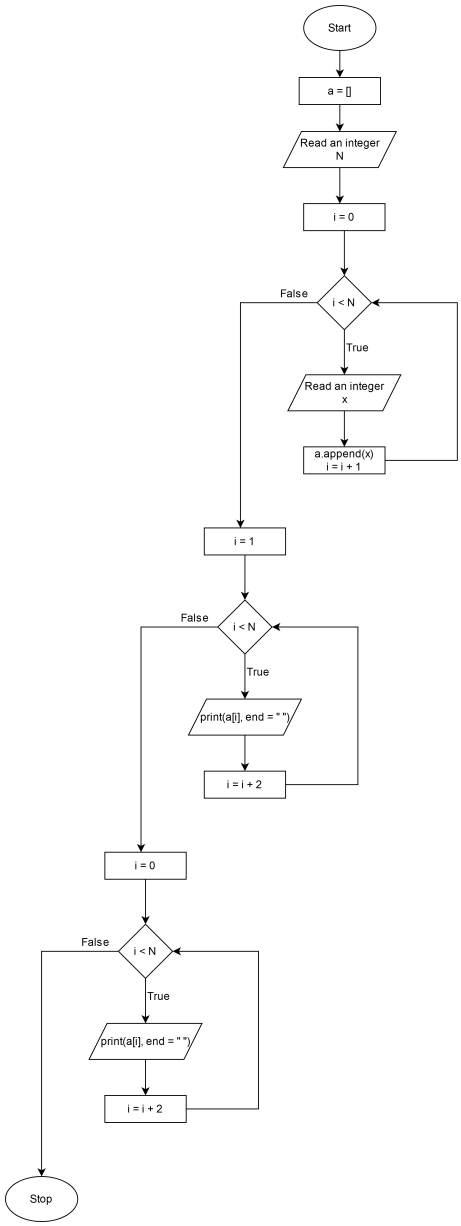

Figure 4: Sample flowchart for Q4

Q 5. Draw a flowchart that takes as input

1. an integer N
2. N integers denoting an array A
3. N integers denoting an array B

and prints N integers, denoting an array C, which is obtained by adding the corresponding elements of A and B. For example for the sample input, 2 1 3 4 5 the output should be 5 8

Ans. We can simple read the arrays A, B and print their sum. This is shown in Fig 5. We need not explicitly construct the array C, since only the output is required, however it is not wrong to do so.

Grading. Any flowchart that prints the given output for all valid inputs is given 2. Solutions that don't explicitly store the array B and just print after reading them is also correct. It is also allowed to explicitly construct the array C, and use "print C" in the flowchart. Any errors in the flowchart results in 1, if there are any errors in the logic then 0.

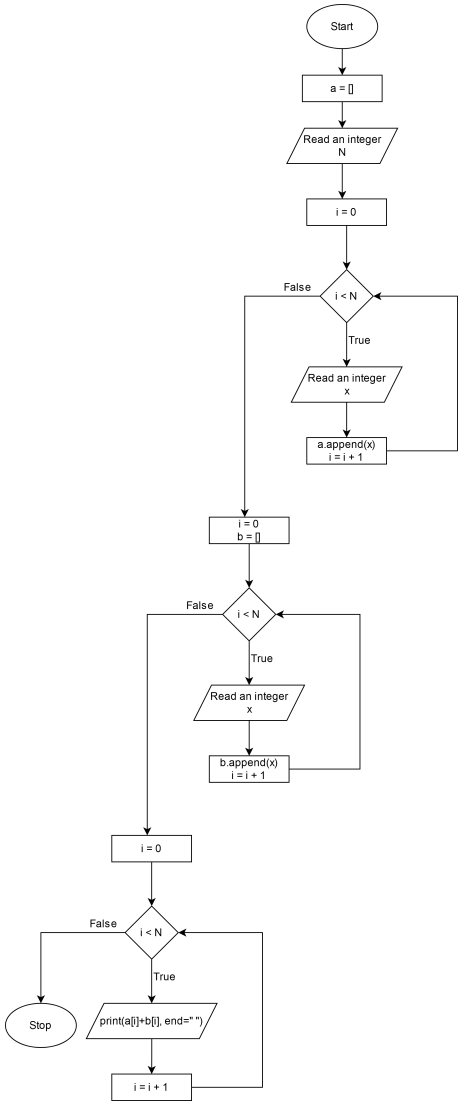

Figure 5: Sample flowchart for Q5

Q 6. Draw a flowchart that takes as input the coefficients of a polynomial and a value x, and evaluates the value of the polynomial at x. Specifically take as input

1. an integer N denoting the number of coefficients of the polynomial
2. N integers, the coefficients, `P[0]`, `P[1]`, ... `P[n-1]`
3. an integer x

and print the value of $P[0] + P[1]x + P[2]x^2 + \ldots P[n-1]x^{n-1}$
For example for the input 3 1 2 3 5 The output should be 86 because, the polynomial given is $P(x) = 1 + 2x + 3x^2$ and
$P(5) = 1 + 2 * 5 + 3 * 5^2 = 1 + 10 + 75 = 86$

Ans. The sample solution given in Fig 6 evaluates the polynomial represented by the last i coefficients in the i^{th} iteration.
Let $Q_i(x) = P_i + P_{i+1}x + \ldots P_{n-1}x^{n-1-i}$ Then $Q_i(x) = xQ_{i+1}(x) + P_i$, what we require is $Q_0(x)$ and $Q_{n-1}(x) = P_{n-1}$ using this formula we can easily evaluate the polynomial by iterating through the coefficients in reverse order and using only $n-1$ multiplications and n additions.
Alternate solutions like using `x ** y` to calculate x^y are also correct.

Grading. If the polynomial is correctly evaluated then 2, if there are any errors in the flowchart then 1, otherwise 0.

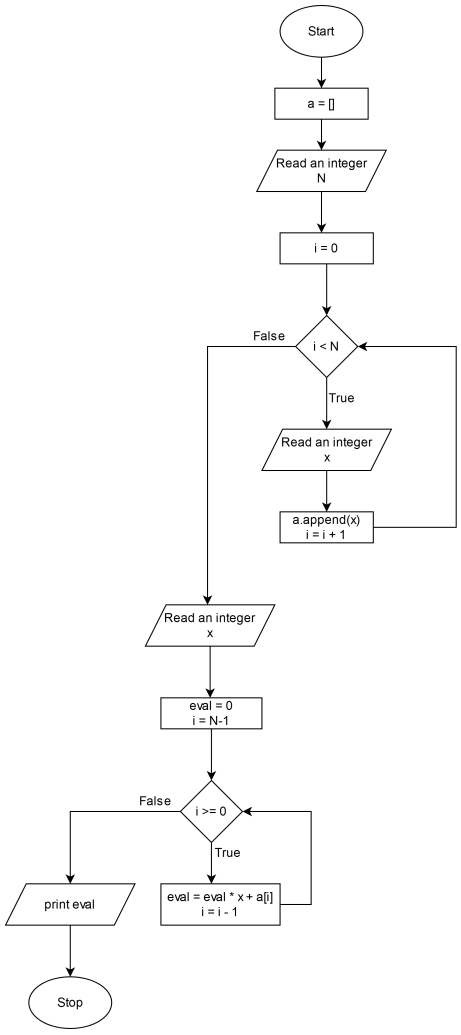

Figure 6: Sample flowchart for Q6

Q 7. Draw a flowchart that takes as input the coefficients of a polynomial and prints all the possible number of positive real zeros that it can have (in increasing order) as predicted by Descartes' rule of signs. (Read the rule from the wikipedia page and implement it in the flowchart).
Specifically the input should be

1. A positive integer N, denoting the number of coefficients
2. N integers denoting the coefficients. If the numbers taken as input are P[0], P[1], ... P[n−1], then the polynomial is $P[0] + P[1]x + P[2]x^2 + ...P[n-1]x^{n-1}$

For example for the input 4 -1 -1 1 -1 the output should be 0 2 because the polynomial is $-1 - x + x^2 - x^3$, the number of changes in sign is 2, so the number of positive real roots can be either 0 or 2 according to the rule of signs, so the output should be 0 2

Ans. Note that coefficients that are 0 must be ignored, that is if there are any coefficients that are 0, then they must be ignored and the adjacent elements must be non-zero. Some corner cases are that the polynomials $-1 - x^2, 1 + x^2, -1 - x^3, 1 + x^3$ all have 0 changes in sign. The polynomial $1 - x^3$ has 1 change in sign.
One way to implement this is to maintain a variable last_sign that stores the sign (1 if positive and -1 if negative) of the last **non-zero** coefficient that is read. Initially the value of last_sign should be 0.
Then we should update last_sign only when the current coefficient is not zero. This is shown in Fig **??**. Once the number of sign changes is correctly identifies, we can iterate from sign_changes%2 up to the number of sign changes, incrementing by 2 each time and printing the coefficients in that order.

Grading. Any flowchart that give the correct output for all valid inputs give 2, if there are any errors in the flowchart but the logic is correct then 1.

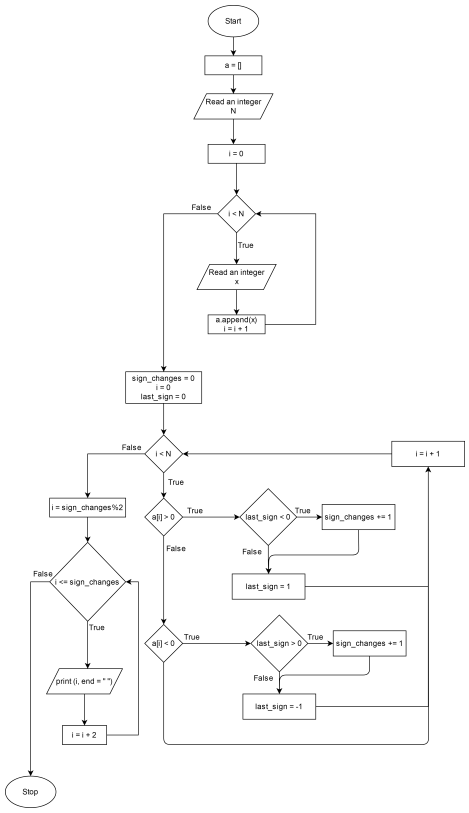

Figure 7: Sample flowchart for Q7