CSCE 5150 Analysis of Computer Algorithms
Assignment 7

Uday Bhaskar Valapadasu
11696364

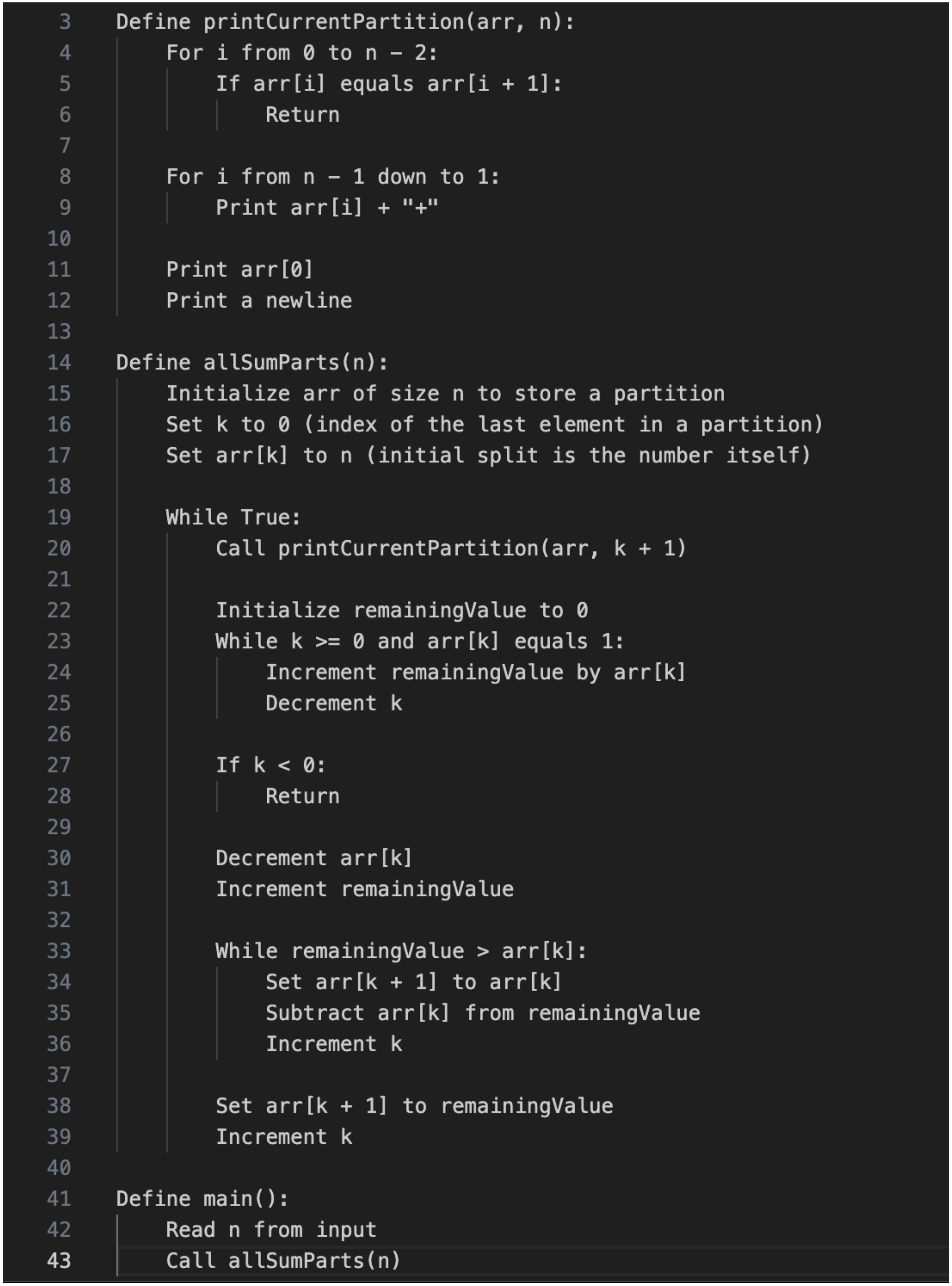
State your answers legibly and concisely. Your solutions will be graded on correctness, elegance, clarity, and originality. Please remember that although group work is permitted, the work handed in must be in your own words.

Design a backtracking algorithm that inputs a natural number n, and outputs all the groups of ASCENDING positive numbers can be summed to give n. Pseudocode is sufficient. An implementation of this algorithm is not necessary.

For example, if n = 6, the output should be
6
1+5
2+4
1+2+3
and if n = 10, the output should be
10
1+9
2+8
3+7
1+2+7
4+6
1+3+6
1+4+5
2+3+5
1+2+3+4

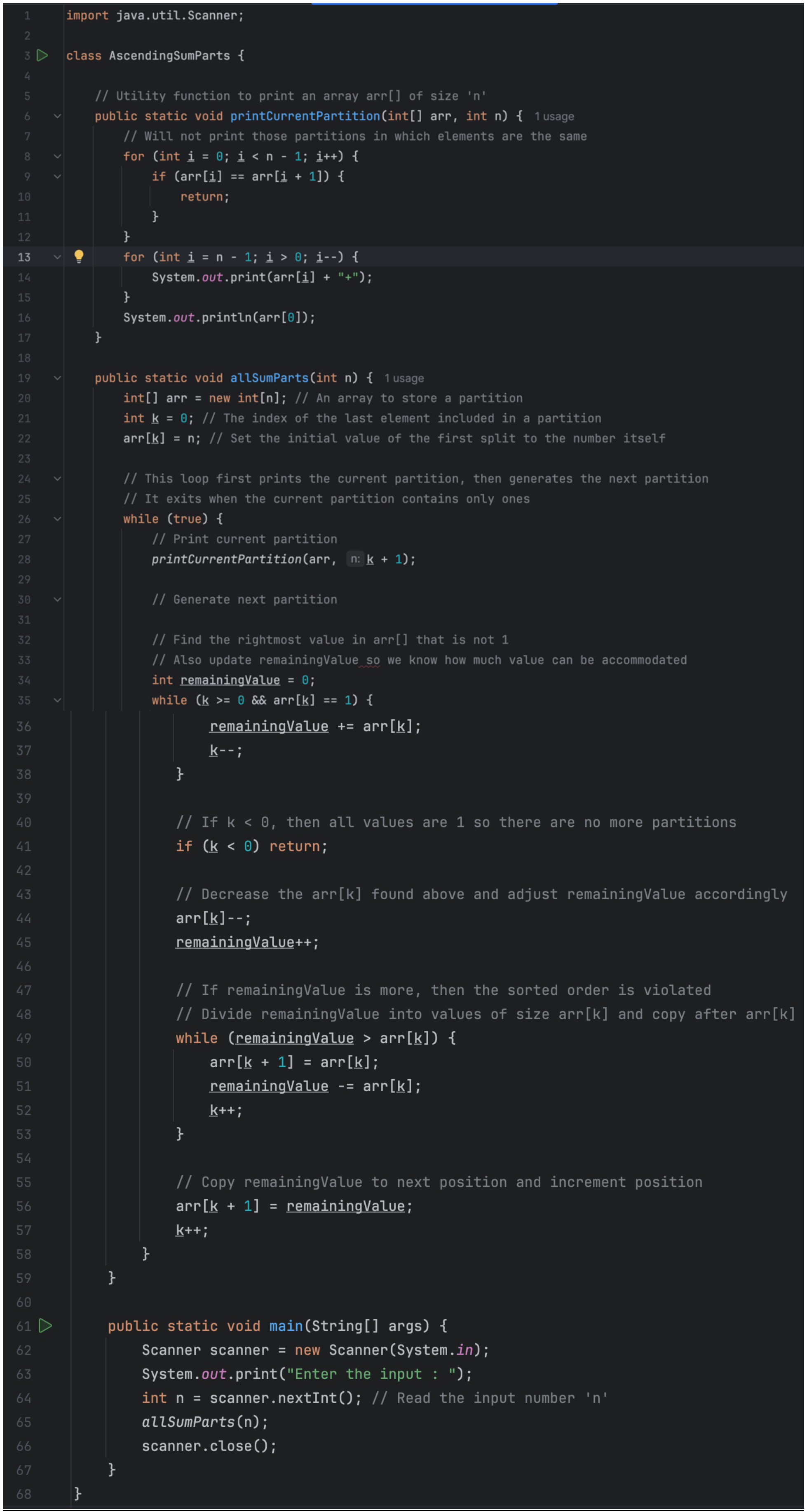
Hint: Store the terms of the current group of ascending positive numbers in an array $A[1..n]$. Backtrack through all possibilities using a variant of the generalized string algorithm in which term $A[i]$ cycles through all values from $A[i + 1] - 1$ to 1.

Pseudocode: Ascending Sum Parts



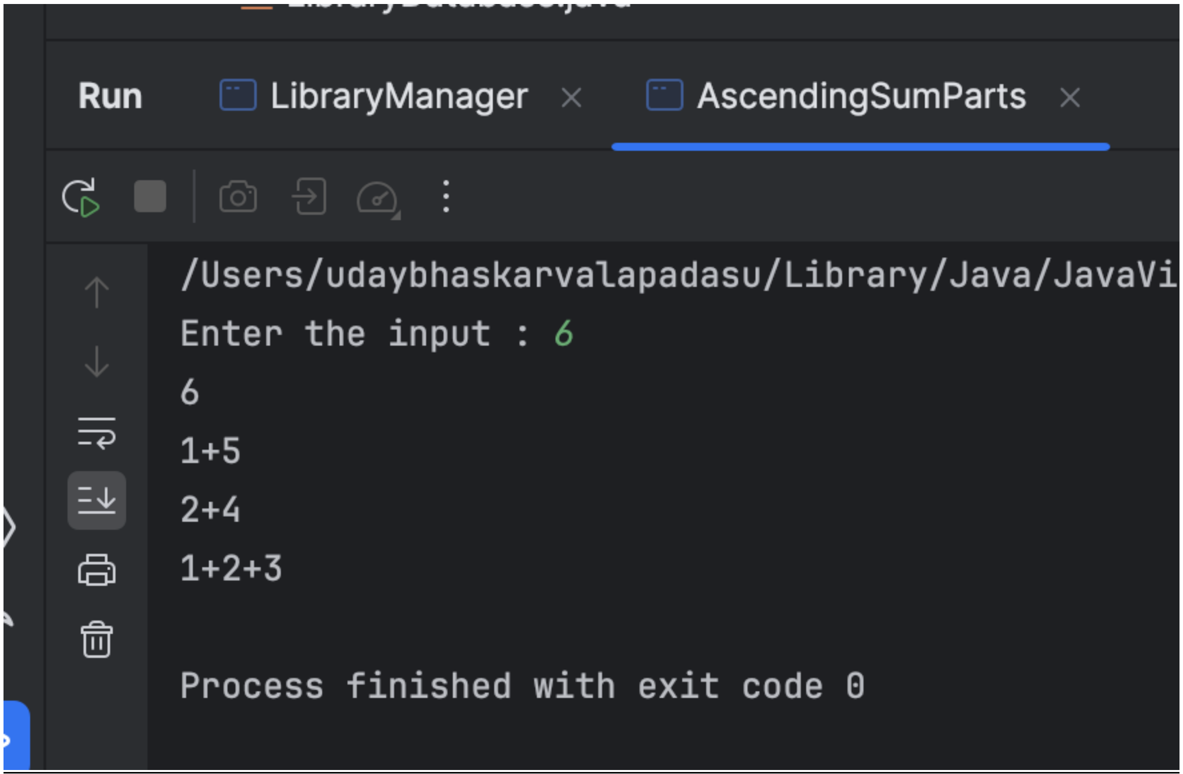
```
Define printCurrentPartition(arr, n):
    For i from 0 to n - 2:
        If arr[i] equals arr[i + 1]:
            Return

    For i from n - 1 down to 1:
        Print arr[i] + "+"

    Print arr[0]
    Print a newline

Define allSumParts(n):
    Initialize arr of size n to store a partition
    Set k to 0 (index of the last element in a partition)
    Set arr[k] to n (initial split is the number itself)

    While True:
        Call printCurrentPartition(arr, k + 1)

        Initialize remainingValue to 0
        While k >= 0 and arr[k] equals 1:
            Increment remainingValue by arr[k]
            Decrement k

        If k < 0:
            Return

        Decrement arr[k]
        Increment remainingValue

        While remainingValue > arr[k]:
            Set arr[k + 1] to arr[k]
            Subtract arr[k] from remainingValue
            Increment k

        Set arr[k + 1] to remainingValue
        Increment k

Define main():
    Read n from input
    Call allSumParts(n)
```

Implementation in JAVA

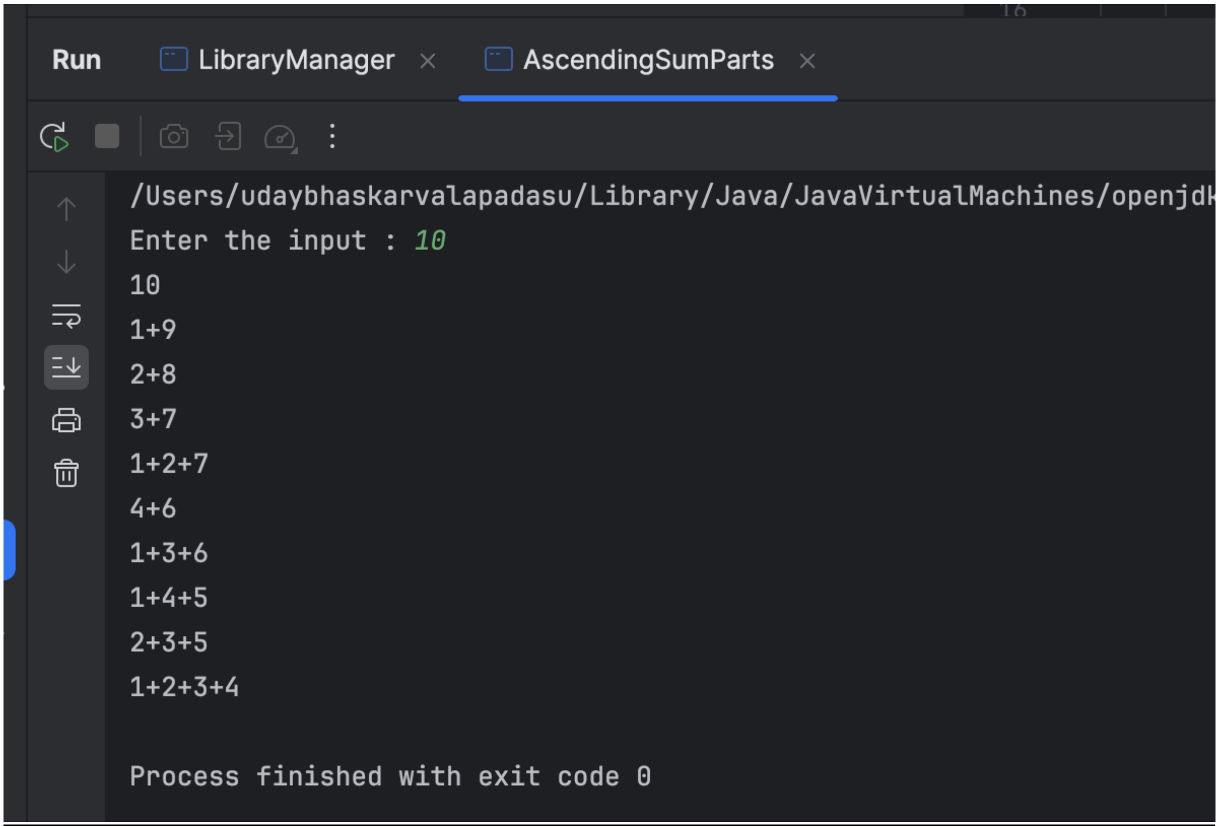


```java
import java.util.Scanner;

class AscendingSumParts {

    // Utility function to print an array arr[] of size 'n'
    public static void printCurrentPartition(int[] arr, int n) {
        // Will not print those partitions in which elements are the same
        for (int i = 0; i < n - 1; i++) {
            if (arr[i] == arr[i + 1]) {
                return;
            }
        }
        for (int i = n - 1; i > 0; i--) {
            System.out.print(arr[i] + "+");
        }
        System.out.println(arr[0]);
    }

    public static void allSumParts(int n) {
        int[] arr = new int[n]; // An array to store a partition
        int k = 0; // The index of the last element included in a partition
        arr[k] = n; // Set the initial value of the first split to the number itself

        // This loop first prints the current partition, then generates the next partition
        // It exits when the current partition contains only ones
        while (true) {
            // Print current partition
            printCurrentPartition(arr, k + 1);

            // Generate next partition

            // Find the rightmost value in arr[] that is not 1
            // Also update remainingValue so we know how much value can be accommodated
            int remainingValue = 0;
            while (k >= 0 && arr[k] == 1) {
                remainingValue += arr[k];
                k--;
            }

            // If k < 0, then all values are 1 so there are no more partitions
            if (k < 0) return;

            // Decrease the arr[k] found above and adjust remainingValue accordingly
            arr[k]--;
            remainingValue++;

            // If remainingValue is more, then the sorted order is violated
            // Divide remainingValue into values of size arr[k] and copy after arr[k]
            while (remainingValue > arr[k]) {
                arr[k + 1] = arr[k];
                remainingValue -= arr[k];
                k++;
            }

            // Copy remainingValue to next position and increment position
            arr[k + 1] = remainingValue;
            k++;
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the input : ");
        int n = scanner.nextInt(); // Read the input number 'n'
        allSumParts(n);
        scanner.close();
    }
}
```

Output:



```
/Users/udaybhaskarvalapadasu/Library/Java/JavaVi
Enter the input : 6
6
1+5
2+4
1+2+3

Process finished with exit code 0
```



```
/Users/udaybhaskarvalapadasu/Library/Java/JavaVirtualMachines/openjdk
Enter the input : 10
10
1+9
2+8
3+7
1+2+7
4+6
1+3+6
1+4+5
2+3+5
1+2+3+4

Process finished with exit code 0
```