LAB NR. 8
C PROGRAMMING

December 6, 2023

Problem 1:

Write a program to reverse an array using recursion.

a) Declare an array of N integers. Use the preprocessor `#define` directive for the size of the array. Fill it with pseudorandom numbers.

b) Define the recursive procedure, that reverses the array by swapping the last with the first element, and then recursively swapping the remaining subarray. To this end, define the function `void reverseArray(int *inputArray, int leftIndex, int rightIndex)`. The function should stop the recursion, whenever `leftIndex >= rightIndex`.

c) Print your original, and the reversed array. Test your results for both even, and for odd N.

Problem 2:

Using the recursion, show all different ways to represent an integer N as sum of non-zero natural numbers. For example, for $N = 3$, you have the following possibilities: 3, $2 + 1$, $1 + 2$, $1 + 1 + 1$.

Problem 3:

Pascal's triangle is a triangular array, useful for calculating the binomial coefficients, $\binom{n}{k}$, that are used in expanding binomials raised to powers, combinatorics and probability theory.

$$\begin{array}{ccccccccc}
 & & & & \binom{0}{0} & & & & \\
 & & & \binom{1}{0} & & \binom{1}{1} & & & \\
 & & \binom{2}{0} & & \binom{2}{1} & & \binom{2}{2} & & \\
 & \binom{3}{0} & & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3} & \\
\binom{4}{0} & & \binom{4}{1} & & \binom{4}{2} & & \binom{4}{3} & & \binom{4}{4}
\end{array}$$

Evaluating the values of the binomial coefficients, you get the following pattern,

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1
\end{array}$$

The number of the entries in each row is increased by one, as we move down. Each number in the triangle, is constructed by adding the number above it and to the left, with the number above it and to the right. The blank entries as treated as 0. Using the recursion, implement the function that computes the Pascal's triangle. Print your result.

Problem 4:

Implement the binary search algorithm, to find an integer in a sorted array.

a) Declare an array of N integers. Use the preprocessor `#define` directive for the size of the array. Let the first element be 0. Fill the array, by assigning to each element the value of the previous element plus a pseudorandom, nonnegative number.

b) Given a sorted array of N integers, write a function to recursively search for a given element x in and return the index of x in the array, or -1 if the element is not found. To this end, divide the interval into two halves. If the value of the element is less than the item in the middle of the interval, search further in the lower half of the interval, otherwise, search it in the upper half.

c) Test your results, searching for randomly picked element from the array.