# Exercise Problems - 12

## Functions - 3

1. Write a program to take $n$ integers as input from the user and output the element that is repeated the maximum number of times. In your program, write the following functions:

   - `void inputArray(int *x, int n)` - for reading n integers from terminal as elements of an array with starting address $x$
   - `int getFrequency(int *x, int n, int k)` - for returning the number of times $k$ is occuring in an array of $n$ elements with starting address $x$.
   - `int getMaxPosn(int *x, int n)` - for returning the index of the maximum element in an array of $n$ elements with starting address $x$ (if a tie occurs, return the index of the first occurrance of the maximum value).

   In your `main` function, declare two arrays, `a[]` for storing the elements and `freq[]` for storing the frequencies of elements in `a[]`. First, call `inputArray()` function to read elements into the array. Then, for each element `a[i]` of `a[]`, compute the number of times it is appearing in `a[]` by calling the function `getFrequency()` with appropriate parameters and store the return value in `freq[i]`. After this, call the function `getMaxPosn()` to compute the position of the maximum value stored in the array `freq[i]`. Using this result, complete your `main` function.

2. In this exercise, we will rewrite the program for selection sort using a recursive function for sort. Define the following functions in the same way as we discussed in class.

   - `void inputArray(int *x, int n)` - for reading n integers from terminal as elements of an array with starting address x
   - `void printArray(int *x, int n)` - for outputing n elements of an array with starting address x
   - `int getMaxPosn(int *x, int n)` - for returning the index of the maximum element among x[0] to x[n-1]
   - `void swap(int *x, int *y)` - for exchaning the values stored at locations pointed to by x and y

   Now, define a recursive function `selectionSort(int *x, int n)` for sorting the n element array with starting address x, as follows: If $n > 1$, call function `getMaxPosn()` to find out the position of the maximum element in the $n$ element array $x$; store the result in variable `posn`; call `swap` function to exchange the values of `x[posn]` and `x[n-1]` to move the maximum element to the last position. After this, recursively sort the first $n - 1$ elements in the array $x$, by calling `selectionSort`() with suitable parameters. Handle the base case $n = 1$ appropriately. Prove that your program is correct.

3. Write a program that takes an intger $n$, followed by elements of four square matrices of dimension `n` $\times$ `n` as input from the user and outputs the matrix which is the sum of all the four input matrices. Define functions

   - `void inputMatrix(int (*x)[],int n)`
   - `void printMatrix(int (*x)[],int n)`
   - `void addMatrices(int (*x)[],int (*y)[],int (*z)[], int n)`

   and use them in your `main` function.

4. Write a program that takes a string as input from the user and outputs a string in which the characters are reversed. Write a function `void reverse(char *source, char *target)` for reversing the string with starting address `source` and store the resultant string in an array of starting address `target`. The string with starting address `source` should be the same after the function call as it was before. In your `main()` function, take input from the user and do the necessary null termination of the input string and size checks before calling `reverse()` and finally print the reversed string.

   Can you do implement the function `reverse()` without using any other variables other than `source` and `target`? Recall that the modifications done to the parameters `source` and `target` are no more effective after returning from a function call, but indirect modifications made to contents of locations pointed to by these pointers are retained even after the function call.