

CMPE 252 - C Programming, Spring 2021

Lab 2

In this lab, you will use the following functions to be implemented for the prelab.

```
void readInput(int arr[], int *nPtr); // reads numbers from the standard input into  
// arr, and stores the number of read elements in the memory cell pointed to by nPtr  
void printNumbers(const int arr[], int n); // prints the elements in arr[0..(n-1)]
```

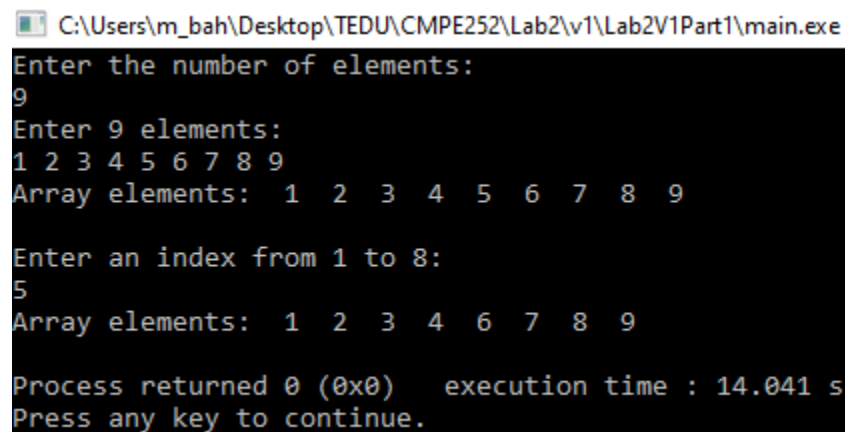
Part 1 (30 points)

Implement the following function in skeleton code `lab2part1.c`:

```
// Precondition: Let n represent size of arr. ind is in range [1 (n-1)].  
// Shift all array elements starting from index ind until end of the array to one  
// position left.  
// Notice that, as a result, the element at index ind-1 is removed and the number of  
// elements in the array is decreased by one.  
// Size of arr is pointed to by np.  
void shiftLeft(int arr[], int *np, int ind);
```

Your task in this part to fill in the missing function definitions in skeleton code `lab2part1.c`.
`main` function will stay as it is.

Sample Run:



```
C:\Users\m_bah\Desktop\TEDU\CMPE252\Lab2\v1\Lab2V1Part1\main.exe  
Enter the number of elements:  
9  
Enter 9 elements:  
1 2 3 4 5 6 7 8 9  
Array elements: 1 2 3 4 5 6 7 8 9  
  
Enter an index from 1 to 8:  
5  
Array elements: 1 2 3 4 6 7 8 9  
  
Process returned 0 (0x0)   execution time : 14.041 s  
Press any key to continue.
```

Part 2 (35 points)

Implement the following function in skeleton code `lab2part2.c`:

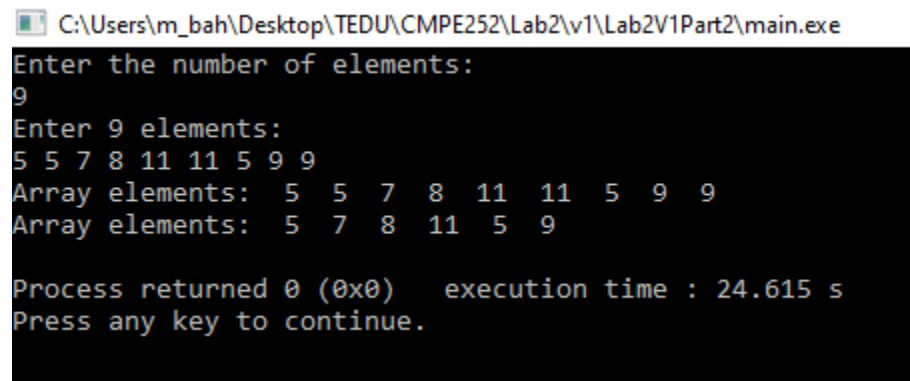
```
// Eliminate consecutive duplications in arr by shifting its elements to left.  
// Update size of arr (which is pointed to by np) accordingly.  
void removeConsecutiveDuplications(int arr[], int *np);
```

Restriction: Use `shiftLeft` function while implementing `removeConsecutiveDuplications` function.

Your task in this part to fill in the missing function definitions in skeleton code `lab2part2.c`.

`main` function will stay as it is.

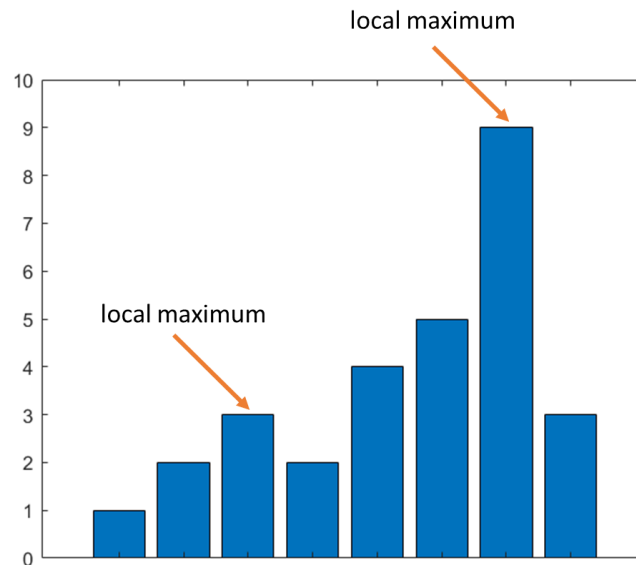
Sample Run:



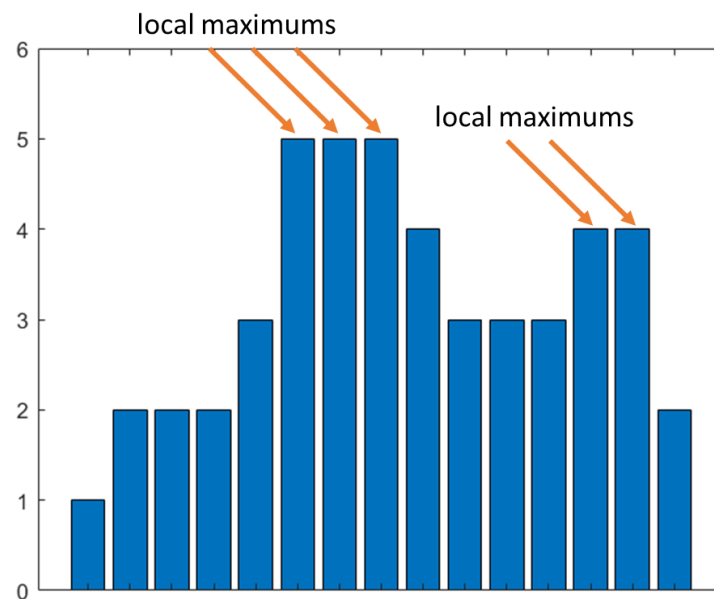
```
C:\Users\m_bah\Desktop\TEDU\CMPE252\Lab2\v1\Lab2V1Part2\main.exe  
Enter the number of elements:  
9  
Enter 9 elements:  
5 5 7 8 11 11 5 9 9  
Array elements: 5 5 7 8 11 11 5 9 9  
Array elements: 5 7 8 11 5 9  
  
Process returned 0 (0x0)   execution time : 24.615 s  
Press any key to continue.
```

Part 3 (35 points)

Given a sequence of values, a local maximum refers to a value which is greater than its left and right neighbors. For example, considering the sequence 1 2 3 2 4 5 9 3, local maximums are 3 and 9. See the figure below.



If the sequence includes consecutive duplicate elements, then a value is a local maximum if it is greater than the first different value at its left and the first different value at its right. For example, considering the sequence 1 2 2 2 3 5 5 5 4 3 3 3 4 4 2, local maximums are 5, 5, 5, 4, 4. See the figure below.




Implement the following function in skeleton code `lab2part3.c`:

```
// Finds local maximums of the sequence of n values in arr and outputs them using  
// localMaxArr array where sp points to the memory cell storing size of localMaxArr.  
void findLocalMax(const int arr[], int n, int localMaxArr[], int *sp);
```

Your task in this part to fill in the missing function definitions in skeleton code `lab2part3.c`.
`main` function will stay as it is.

Sample Run:

 C:\Users\m_bah\Desktop\TEDU\CMPE252\Lab2\v1\Lab2V1Part3\main.exe

```
Enter the number of elements:  
9  
Enter 9 elements:  
1 2 3 4 3 5 10 11 9  
Array elements: 1 2 3 4 3 5 10 11 9  
Local maximum array:  
Array elements: 4 11  
  
Process returned 0 (0x0)   execution time : 48.120 s  
Press any key to continue.
```