

1 Insertion Sort

Implement the following function:

```
function x = InsertionSortInplace(x)
% Sort x in ascending order using the insertion sort algorithm.
% Sort in-place, i.e., without creating another vector.
% Perform the insert process in-line, i.e., no subfunction.
% x is a column vector.
```

For your reference, below is the InsertionSort function we discussed in lecture.

```
function [x,TotalC,TotalS] = InsertionSort(x)
% Sort x in ascending order using insertion sort algorithm.
% x is a column n-vector.
% TotalC is the total number of required comparisons.
% TotalS is the total number of required swaps.

n = length(x); TotalC = 0; TotalS = 0;
for k = 2:n
    [x(1:k),C,S] = Insert(x(1:k));
    TotalC = TotalC + C; TotalS = TotalS + S;
end

function [x,C,S] = Insert(x)
% Pre: x is a column m-vector with x(1:m-1) sorted.
% Post: x is sorted in ascending order by applying the insert process.
% C is the number of required comparisons.
% S is the number of required swaps.
m = length(x); S = 0;
k = m-1;
while k>=1 && x(k)>x(k+1)
    t = x(k+1); x(k+1) = x(k); x(k) = t;
    S = S+1;
    k = k-1;
end
C = S+1
```

2 Merge Sort

The code for functions `mergeSort` and `merge` are shown below. What is the output when you run the execute the following statements?

```
a= [4 1 6 3 2 9 5 7 6 0];  
b= mergeSort(a);
```

Trace the execution carefully. Note that `mergeSort` is *recursive*, so multiple calls of `mergeSort` can be open at the same time. Ask your section instructor if you have any questions!

```
function y = mergeSort(x)  
% x is a vector.  
% y is a vector consisting of the values in x sorted from smallest to largest.  
  
n = length(x)           % length of vector x is displayed  
if n==1  
    y = x;  
else  
    m = floor(n/2);  
    % Sort the left half..  
    yL = mergeSort(x(1:m)) % values displayed are the values returned by this call of mergeSort  
    % Sort the right half..  
    yR = mergeSort(x(m+1:n)) % values displayed are the values returned by this call of mergeSort  
    % Merge..  
    y = merge(yL,yR)       % values displayed are the values returned by this call of merge  
end
```

```
function z = merge(x,y)  
% x and y are sorted vectors and z is their merge.  
% x(1) <= x(2) <= ... <= x(nx)  
% y(1) <= y(2) <= ... <= y(ny)  
% z is a sorted vector with length nx+ny and comprises all the values in x and y:  
% z(1) <= z(2) <= ... <= z(nx+ny)  
  
nx = length(x); ny = length(y);  
z = zeros(1, nx+ny);  
ix = 1; iy = 1; iz = 1;  
while ix<=nx && iy<=ny % x and y have not been exhausted  
    if x(ix) <= y(iy)  
        z(iz)= x(ix); ix=ix+1; iz=iz+1;  
    else  
        z(iz)= y(iy); iy=iy+1; iz=iz+1;  
    end  
end  
while ix<=nx % copy any remaining x-values  
    z(iz)= x(ix); ix=ix+1; iz=iz+1;  
end  
while iy<=ny % copy any remaining y-values  
    z(iz)= y(iy); iy=iy+1; iz=iz+1;  
end
```