Suggested Solutions to HW #2

5.6 Modify algorithm Maximum_Consecutive_Subsequence (Fig. 5.9) such that it finds the actual subsequence and not only the sum.

Solution. (Ming-Hsien Tsai, and modified by Yi-Wen Chang)

```
Algorithm Max_Conseq_Subseq(X, n);
begin
    $Global\_Max := 0$;
    $Suffix\_Max := 0$;
    $Suffix\_Start\_Index := 1$;
    $Global\_Start\_Index := 0$;
    $Global\_End\_Index := 0$;
    for $i := 1$ to $n$ do
        if $x[i] + Suffix\_Max > Global\_Max$ then
            $Suffix\_Max := Suffix\_Max + x[i]$;
            $Global\_Max := Suffix\_Max$;
            $Global\_Start\_Index := Suffix\_Start\_Index$;
            $Global\_End\_Index := i$;
        else if $x[i] + Suffix\_Max > 0$ then
            $Suffix\_Max := Suffix\_Max + x[i]$;
        else
            $Suffix\_Max := 0$;
            $Suffix\_Start\_Index := i + 1$;
end
```

□

5.12 Let $x_1, x_2, \cdots, x_n$ be a sequence of real numbers (not necessarily positive). Design an $O(n)$ algorithm to find the subsequence $x_i, x_{i+1}, \cdots, x_j$ (of consecutive elements) such that the product of the numbers in it is maximum over all consecutive subsequences. The product of the empty subsequence is defined as 1.

Solution. (Wen-Chin Chan, and modified by Jinn-Shu Chang)

```
Algorithm Maximum_Consecutive_Subsequence(X,n);
begin
    $Global\_Max := 1$;
    $Suffix\_Pos := 1$;
    $Suffix\_Neg := 1$;
    for i := 1 to n do
       if $X[i] > 0$ then
            if $Suffix\_Pos \times X[i] > Global\_Max$ then
                 $Global\_Max := Suffix\_Pos \times X[i]$;
            $Suffix\_Pos := Suffix\_Pos \times X[i]$;
            $Suffix\_Neg := Suffix\_Neg \times X[i]$;
            if $Suffix\_Pos < 1$ then
                 $Suffix\_Pos := 1$;
            if $Suffix\_Neg \geq 0$ then
                 $Suffix\_Neg := 1$;
       else if $X[i] < 0$ then
            if $Suffix\_Neg \times X[i] > Global\_Max$ then
                 $Global\_Max := Suffix\_Neg \times X[i]$;
            $Suffix\_Pos := Suffix\_Pos \times X[i]$;
            $Suffix\_Neg := Suffix\_Neg \times X[i]$;
            $swap(Suffix\_Pos, Suffix\_Neg)$;
            if $Suffix\_Pos < 1$ then
                 $Suffix\_Pos := 1$;
       else /* $X[i] = 0$ */
            $Suffix\_Pos := 1$;
            $Suffix\_Neg := 1$;
end
```

□