Problem A:

Write a function named get_MSS which gets an array of integers and returns sum of the subarray (possibly empty) with maximum sum.

Example:

```
> (get_MSS '(1 2 -10 4 5))
9
> (get_MSS '(1 2 -10 9 2 -1 0))
11
```

Problem B:

Let's define an environment Env a mapping from variables to values. Consider a list of pairs in which each pair's first element is a string (variable's name) and second element is an integer (variable's value) as a presentation of an environment in racket.

For example consider an environment E a mapping from A to 2 and from B to 3, following list is the presentation of E:

```
'(("A" . 2) ("B" . 3))
```

Write a function named $merge\_envs$ which gets two environments $E1$ and $E2$ and returns a merged environment $mergedEnv$.

For all variables like v in $E1$ or $E2$, $mergedEnv[v]$ is $E2[v]$ if v has a mapping in $E2$, otherwise $mergedEnv[v]$ is $E1[v]$.

Example:

```
> (list (cons "A" 2) (cons "B" 3))
'(("A" . 2) ("B" . 3))
> (list (cons "B" 1) (cons "C" 4))
'(("B" . 1) ("C" . 4))
> (merge_envs (list (cons "A" 2) (cons "B" 3)) (list (cons "B"
1) (cons "C" 4)))
'(("A" . 2) ("B" . 1) ("C" . 4))
```

Problem C:

Consider a right-regular grammar G, all G's rules are in following form:

$< nonterminal > ::= < terminal >< nonterminal > | \varepsilon$

$< terminal > ::= a | b | c | d$

$< nonterminal > ::= S|A|B|C|D$

A rule is stored in following structure:

```
(struct rule (left_nonterminal terminal right_nonterminal)
#:transparent)
```

For example following rule:

$S ::= aA | \varepsilon$

Is stored like below:

```
(rule "S" "a" "A")
```

Write a function named $is\_derived$ which gets a string and a list of rules of a described grammar and returns if the string is derived from the input grammar.

It is guaranteed that:

- There are at most four rules.
- String length at most ten.
- There is at least one rule with left non-terminal named S.

Example:

```
> (define gr (list (rule "S" "s" "A") (rule "A" "a" "A") (rule
"A" "b" "A")))
> (is_derived gr "sabb")
#t
> (is_derived gr "abb")
#f
```

You can check out this link for templates, tests and updates.