National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2016/2017

Recitation 8
Mutable Data Structures

Source Week 8

1. `x = ⟨expression⟩;`
   Assignment statement. Assigns the result of evaluating ⟨*expression*⟩ as new value to variable `x`; returns the value.
2. `set_head(p, x)`
   Sets the head (first component) of the pair `p` to be `x`; returns `undefined`.
3. `set_tail(p, x)`
   Sets the tail (second component) of the pair `p` to be `x`; returns `undefined`.

Problems:

1. Consider the following implementation of a stack object.

```
function make_stack() {
    var stack = pair("stack", []);
    return stack;
}
```

   Note that the stack is represented by a pair whose head is a tag `"stack"`. The stack is represented by a list, which of course is initially empty.

   We can create a new stack using the program:

```
var my_stack = make_stack();
```

   (a) Add a function called `is_empty` which returns `true` iff the stack is empty. The function `is_empty` does not change the stack.

```
function is_empty(stack) {
    // complete the function here
}
```

   (b) Add a function called `clean` which empties the stack of any elements it may contain. The function `clean` should change the stack *destructively* (i.e. the previous state is "destroyed"), and return `undefined`.

```
function clean(stack) {
    // complete the function here
}
```

(c) Add a function called `peek` which returns the top element of the stack, leaving the stack unchanged. If the stack is empty, signal a "stack underflow" error.

```
function peek(stack) {
    // complete the function here
}
```

(d) Add a function called `push` which allows an element to be added to the top of the stack. The function `push` should change the stack destructively, and return `undefined`.

```
function push(stack, x) {
    // complete the function here
}
```

(e) Add a function called `pop` which removes and returns the top element of the stack. The function `pop` should change the stack destructively, and return `undefined`.

```
function pop(stack) {
    // complete the function here
}
```

2. Write a function called `push_all` which takes a stack and a list and pushes all the elements of the list onto the stack. The function `push_all` should change the stack destructively, and return `undefined`.

```
function push_all(stack, lst) {
    // complete the function here
}
```

3. Write a function called `pop_all` which takes a stack and pops elements off it until it becomes empty, adding each element to the output list. The function `pop_all` should change the stack destructively.

```
function pop_all(stack) {
    // complete the function here
}
```

4. Implement the `reverse` function only using the stack operations you have implemented so far.

```
function reverse(lst) {
    // complete the function here
}
```