Javascript Closures

R3.2/3.3

heap memory Make()(10) stack memory

```
function make() {
    var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
     console.log(x.toString()+ " does
    something Meaningful");
     else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(10);
(make())(4);
```

Make()(10) Counter=0

stack memory

Mair

```
action make(){
    var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
    console.log(x.toString()+ " does
something Meaningful");
     else
     console.log(x.toString() + " does
    not do something meaningful");
    counter++;
(make())(10);
(make())(4);
```

Make()(10) Counter=0

stack memory

/lain

```
function make(){
   var counter=0;
    return function(x){//returns a function which will state whether
     or not x does something meaningful
      if(counter==0)
    console.log(x.toString() + " does
something Meaningful");
      else
      console.log(x.toString()+ " does
     not do something meaningful");
     counter++;
(make())(10);
(make())(4);
```

Function(10)

stack memory

Mair

```
← to
```

```
function make() {
    var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
     console.log(x.toString()+ " does
    something Meaningful");
     else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(10);
(make())(4);
```

Function(10)

stack memory

Main

```
function which
                      11 state whether
    or not x does
                       thing meaningful
     if(counter==0
    console.log(x.toString() + " does
something Meaningful");
     else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(10);
(make())(4);
```

return function(x){//returns a

function make() {

var counter=0;

Make()(4)

stack memory

Mair

```
function make() {
    var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
    console.log(x.toString() + " does
something Meaningful");
      else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(4);
```

Make()(4) Counter=0

stack memory

Main

```
function make (
     var counter
    return function(x){//returns a
function which will state whether
     or not x does something meaningful
      if(counter==0)
    console.log(x.toString()+ " does
something Meaningful");
      else
      console.log(x.toString() + " does
     not do something meaningful");
     counter++;
(make())(10);
(make())(4);
```

Function(4)

stack memory

Main

```
var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
    console.log(x.toString() + " does
something Meaningful");
     else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(4);
```

function make() {

stack memory

```
function make() {
    var counter=0;
    return function(x){//returns a
    function which will state whether
    or not x does something meaningful
     if(counter==0)
     console.log(x.toString()+ " does
    something Meaningful");
     else
     console.log(x.toString()+ " does
    not do something meaningful");
    counter++;
(make())(10);
(make())(4);
```

R3.3

- I found that our example didn't really have a
 way of using functions that were returned as
 values. I just used Function(x) which isn't
 exactly true but I found it did a reasonable job
 of explaining what was going on.
- I personally had trouble with this question because I was unsure of what simple useful purposes this serves.