1. Global lexical environment.

a. Creation

LE { } TDZ : { makearmy : fn
army : [] }

outer : null

b. Execution:

LE { makearmy : fn
army : [f(){alert(j)}, f(x){alert(j);}] } TDZ :

outer : null

2. LE for makearmy

Creation

LE : { } TDZ : shooters : []
i : 0

outer : global

Execution:

LE : { shooters : [f(){ alert(j);
f(x){ alert(j);
]

i : 2 }

outer : global

3. LE for LE of the while loop

a. Loop where $i=0$

<u>creation</u>

LE: { } <u>TDZ</u>: shooter: fn

outer: makearmy $i=0$

<u>Execution:</u>

LE: { shooter: f() { alert(i); } <u>TDZ</u>:

shooters: [f() { alert(i); }]

$i=1$

}

outer: makearmy

b. Loop where $i=1$

<u>Creation:</u>

LE: { } <u>TDZ</u> shooter: fn

outer: makearmy $i=1$

<u>Execution</u>

LE: { shooter: f() { alert(i); } <u>TDZ</u>:

i: 2

shooters: [f() { alert(i); },

f(x) { alert(i); }

]

}

outer: makearmy

LE for army[0]

Creation

LE: { } ID2:

closures global

free variable: i

execution:

LE: { free variable i: 2 } ID2

closure: global

what will army[0] alert?

=> army[0] will return function as:

```
f() {
    alert(i);
}
```

The function will not be executed.

Can you fix the code?

→ using IIFE we can modify it as army[0]();

so that it is executed immediately.

Doing so the alert value will be 2. Also

to ensure alert as 0, define scope variable in while.

How will the diagram change?

→ changes will be in LE of while, where

value of x will be retained on execution

Here, x is a local variable.