Question 1:

global LE Creation = {makeArmy: <fn>, outer: null, this: window} TDZ= {army}

global LE Execution= {army: [<fn>, <fn>], makeArmy: <fn>, outer: null, this: window} TDZ= {}

makeArmy LE Creation = {arguments: {length: 0}, outer: global, this: window} TDZ= {shooters, i}

makeArmy LE Execution = {shooters: [<fn>, <fn>], i: 2, arguments: {length: 0}, outer: global, this: window} TDZ= {}

while LE Creation = {outer: makeArmy, this: window} TDZ= {shooter}

while-0 LE Execution= {shooter: <fn>, outer: makeArmy, this: window} TDZ= {}

while-1 LE Execution= {shooter: <fn>, outer: makeArmy, this: window} TDZ= {}

army[0] LE Creation= {arguments: {length: 0}, outer: while, this: window} TDZ={}

army[0] LE Execution= (closure:(makeArmy LE Execution)) {arguments: {length: 0}, outer: while, this: window} TDZ={}

Fixed code (also in html page):

```
function makeArmy() {
    let shooters = [];
    let i = 0;
    while (i < 2) {
        let j = i;
        let shooter = function () {
            console.log(j);
        };
        shooters.push(shooter);
        i++;
    }
    return shooters;
}
let army = makeArmy();
army.forEach(f => f());
```

Changes in diagram:

Shooter function would not get its argument as i from closure, instead it will get j from while scope.

while LE Creation = {outer: makeArmy, this: window} TDZ= {shooter, j}

while-0 LE Execution= {j: 0, shooter: <fn>, outer: makeArmy, this: window} TDZ= {}

while-1 LE Execution= {j: 1, shooter: <fn>, outer: makeArmy, this: window} TDZ= {}

Question 2:

Look at the html page console/sources or link for the code.

```
function printNumbers(from, to){
    let isSmaller = from <= to;
```

```
    let current = from;
    let timer = setInterval(()=>{
        console.log(current);
        if(current === to){
            clearInterval(timer);
        }
        current += isSmaller ? 1 : -1;
    }, 1000);
}
```

Question 3:

i would be 100000000, and alert would show up after the execution of the loop since setTimeout would be pushed into callback queue and waiting for the call stack to get empty.