SERVER SIDE SMALL TEST 1

VERSION B

Your name:

# Question 1 [18 points]

Consider these two lines of code.

let list = ['hoi', 23, 39.5];

const x = 1;

For each expression below, encircle the answer that the expression produces.

list[1]

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 23 | undefined | An Error |

list[x]

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 23 | undefined | An Error |

list.x

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 23 | undefined | An Error |

list[list['length']]

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 39.5 | undefined | An Error |

list.length

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 3 | undefined | An Error |

list.0

### Your Answer:

|  |  |  |
| --- | --- | --- |
| hoi | undefined | An Error |

# Question 2 [10 points]

Encircle the ***two*** code snippets that would produce an error (if you'd run them).

const list = [1, 2, 3];

list[2] = 4;

const list = [1, 2, 3];

list = 10;

const list = [1, 2, 3];

list.push('hi');

const list = [1, 2, 3];

list = [5, 6];

const list = [1, 2, 3];

console.log(list);

**Question 3 [4 points]**

Encircle the ONE correct statement:

1. A pure function can call *unpure* functions
2. A pure function can read a variable from the global scope
3. A pure function can modify the elements in the provided array argument
4. A pure function can return null

# Question 4 [3 points]

Consider this code fragment

let animal = {

sound: 'growl'

};

let otherAnimal = animal;

otherAnimal.sound = 'tjirp';

console.log(animal.sound);

What is the output of console.log(animal.sound);?

### Your Answer:

|  |  |  |
| --- | --- | --- |
| tjirp | growl | An Error |

# Question 5 [15 points]

For each code snippet below, encircle the correct output on the console.

console.log(x);

var x = 10;

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 10 | undefined | An Error |

console.log(x);

let x = 10;

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 10 | undefined | An Error |

const check = true;

if (check) {

let x = 10;

}

console.log(x);

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 10 | undefined | An Error |

for (var i = 0; i < 11; i++) {

var x = i;

}

console.log(x);

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 10 | undefined | An Error |

function setX(value) {

var x = value;

}

setX(10);

console.log(x);

### Your Answer:

|  |  |  |
| --- | --- | --- |
| 10 | undefined | An Error |

# Question 6 [40 points]

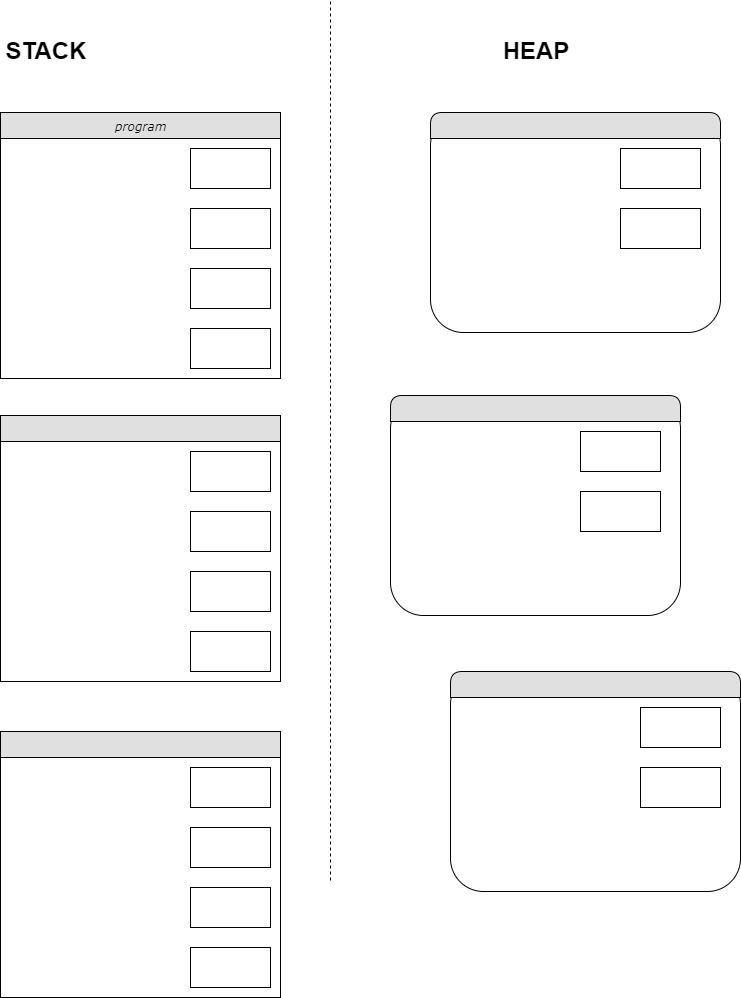
Given the following code:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | const execute = (nrTimes, func) => {  var person = {  name: 'han',  number: 21  };  for (var i = 0; i < nrTimes; i++) {  func(person.name);  }  }  const n = 2;  const sayHiTo = (name) => {  console.log(`Hi ${name}`);  }  execute(n, sayHiTo); |

Complete the memory model diagram on the next page when the program is executing the function execute on line 17 and inside execute, the function func on line 7 is currently being executed. The value of i is currently equal to 1.

### Remarks

* You don’t have to give the stack frames and heap objects a name. This means that you can leave the gray area of all containers in the diagram empty.
* You can leave boxes you don’t need empty.



# Question 7 [10 points]

Given the following code:

const getValidNumber = (number) => {

if (number === null) {

throw new Error('Not Valid');

}

return number;

}

const printNumbers = (list) => {

for (let i = 0; i < list.length; i++) {

console.log(getValidNumber(list[i]));

}

}

try {

printNumbers([10, null, 35.5]);

} catch (err) {

console.log(err.message);

}

Write down the console output when program is run.

### Your Answer:

|  |
| --- |
|  |