

# COMPSCI 220 Midterm 1: Fall 2019

---

Name: \_\_\_\_\_

Spire ID: \_\_\_\_\_

DO NOT OPEN EXAM BOOKLET UNTIL INSTRUCTED TO DO SO.
---

**Time for Exam** 60 minutes

**Instructions** Do not begin your exam until instructed. Please read all rules carefully before beginning your exam. You will have one hour to complete all exam problems to the best of your ability.

**Points** This exam has 8 questions worth 100 points total.

**Question Difficulty** The expected difficulty of each question is marked with \*s. Harder questions have more \*s. This should help you plan your time.

## Rules

1. All electronics (including but not limited to cellphones, tablets, computers, smart watches, and calculators) must be turned off and placed out of sight in your backpack.
2. In order to receive credit for your midterm you *must* write your name and Spire ID on the first page, and your name on all other pages.
3. All answers must fit within the boxes allocated for that question. Any work outside of the box will not be taken into account during grading.
4. There will be no talking or leaving the exam room during testing.
5. We have provided exam notes, which you may write on. You may not use any other material or notes.
6. All work must be your own and compliant with the Universities Academic Honesty Policy. Any exhibition of Academic Dishonesty will be reported to the Academic Honesty Board.

**Question 1** What do the following programs display? *Note that none of these programs produce errors.*

**Part a** (5 points) *Difficulty: ★*

```
function F1(x) {
  if (x === 3) {
    return;
  }
  console.log(x);
  F1(x + 1);
}
F1(1);
```

**Part b** (5 points) *Difficulty: ★★*

```
function app(f, x) {
  return f(x);
}
app(function(z) {
  console.log('1');
  return function(w) {
    console.log('2');
  };
}, function(y) {
  console.log('3');
});
```

**Part c** (5 points) *Difficulty: ★★*

```
let arr = [ ];
let o = { x: 1 };
for (let i = 0; i < 10; i = i + 1) {
  arr.push(o);
}
arr[3].x = 100;
let sum = 0;
for (let i = 0; i < 10; i = i + 1) {
  sum = sum + arr[i].x;
}
console.log(sum);
```

**Part d** (5 points) *Difficulty: \*\*\**

```
function mystery(f, n) {
  if (n === 0) {
    return function(x) { return x; }
  } else {
    return function(x) {
      return mystery(f, n - 1)(f(x, n));
    }
  }
}

function H(x, n) {
  return x + n.toString();
}

console.log(mystery(H, 5)(''))
```

**Part e** (5 points) *Difficulty: \**

```
let X = { a: 10, b: { a: 100 } };
X.b = X;
X.b.a = 200;
console.log(X.a);
```

**Part f** (5 points) *Difficulty: \**

```
let X = { a: 10, b: { a: 100 } };
X.b.a = X;
console.log(X.b.a.a);
```

**Question 2** Consider the following function.

```
function D(x, y) {  
  if (x < y) {  
    console.log('A');  
  }  
  function C() {  
    if (x > 10) {  
      console.log('B');  
    }  
  }  
  if (x + y > 10) {  
    C();  
  }  
}
```

**Part a** (5 points) *Difficulty:* ★

Give a value for  $x$  and  $y$  such that  $D(x, y)$  displays:

A

**Part b** (5 points) *Difficulty:* ★

Give a value for  $x$  and  $y$  such that  $D(x, y)$  displays:

B

**Part c** (5 points) *Difficulty:* ★

Give a value for  $x$  and  $y$  such that  $D(x, y)$  displays:

A

B

**Question 3** (5 points) *Difficulty: \*\**

Consider the following function.

```
function K(f) {  
  f(function(x) {  
    return function() {  
      console.log(x);  
    }  
  });  
}
```

Give a value for  $f$  such that  $K(f)$  displays the output:

220

**Your answer must not use `console.log` or raise errors.**

**Question 4** (5 points) *Difficulty: \**

Consider the following function.

```
function T(f) {  
  f(function(x) { console.log(x); });  
}
```

Give a value for  $f$  such that  $T(f)$  displays the output:

220

**Your answer must not use `console.log` or raise errors.**

**Question 5** Consider the following function.

```
function J(f) {
  f(function(x) {
    if (x < 5) { console.log("A"); }
  },
  function(y, z) {
    if (y < z) { return function() { console.log("B"); }; }
    else { return function() { console.log("C"); } }
  });
}
```

**Part a** (5 points) *Difficulty: \*\**

Give a value for  $f$ , such that  $J(f)$  displays the following output:

A

**Your answer must not use console.log.**

**Part b** (5 points) *Difficulty: \*\**

Give a value for  $f$ , such that  $J(f)$  displays the following output:

B

**Your answer must not use console.log.**

**Part c** (5 points) *Difficulty: \*\**

Give a value for  $f$ , such that  $J(f)$  displays the following output:

C  
B

**Your answer must not use console.log.**

**Question 6** The `map2` function is a variation of `map` that applies its functional argument to corresponding pairs of elements from two arrays. It's type and implementation are shown below:

```
// map2<A,B,C>(f: (x: A, y: B) => C, arr1: A[], arr2: B[]): C[]
// Assumes that arr1.length === arr2.length
function map2(f, arr1, arr2) {
  let r = [ ];
  for (let i = 0; i < arr1.length; i = i + 1) {
    r.push(f(arr1[i], arr2[i]));
  }
  return r;
}
```

The Midterm 1 Notes include an implementation of `map`.

**Part a** (5 points) *Difficulty: ★★*

Re-implement `map` *without any loops or conditionals*, using `map2` as a helper function.

**Part b** (5 points) *Difficulty: \*\**

The following function consumes two arrays of equal length and produces a single array that contains pairs of corresponding values from each input array.

```
// zip:(arr1: A[], arr2: B[]): { first: A, second: B }[]  
function zip(arr1, arr2) {  
  let r = [ ];  
  for (let i = 0; i < arr1.length; i = i + 1) {  
    r.push({ first: arr1[i], second: arr2[i] });  
  }  
  return r;  
}
```

Re-implement `map2` *without any loops or conditionals*, using `zip` as a helper function.



**Question 7** The `filterMap` function shown below combines the mapping and filtering operations:

```
// filterMap<A,B>(f: (x: A) => B, g: (y: B) => boolean, arr: A[]): B[]
function filterMap(f, g, arr) {
  return filter(g, map(f, arr));
}
```

The Midterm 1 Notes include implementations of `map` and `filter`.

**Part a** (5 points) *Difficulty: ★★*

Provide an alternative implementation of `filterMap` that behaves the same as the one given above. However, you should traverse the array exactly once. Feel free to use any JavaScript feature you wish in your code.

**Part b** (5 points) *Difficulty: ★★★★★*

You now have two implementations of `filterMap` that are identical, or are they? Do they truly behave the same on *all* possible inputs? Give values for `f`, `g`, and `arr`, such that `filterMap(f, g, arr)` will 1) throw an exception with one implementation of `filterMap` and 2) return normally with the other. **You may define auxiliary variables (and will need to do so).**

You can write `throw 'error'` to throw an exception in JavaScript. i.e., in either `f` or `g`, you should call `throw 'error'` when the program detects it is running one implementation and not the other. Note that you are just giving values for `f`, `g`, and `arr`, so your answer should not call `filterMap` directly. You should not use `console.log` either.

**Question 8** The `doubleFilter` function shown below applies two filtering functions to an array:

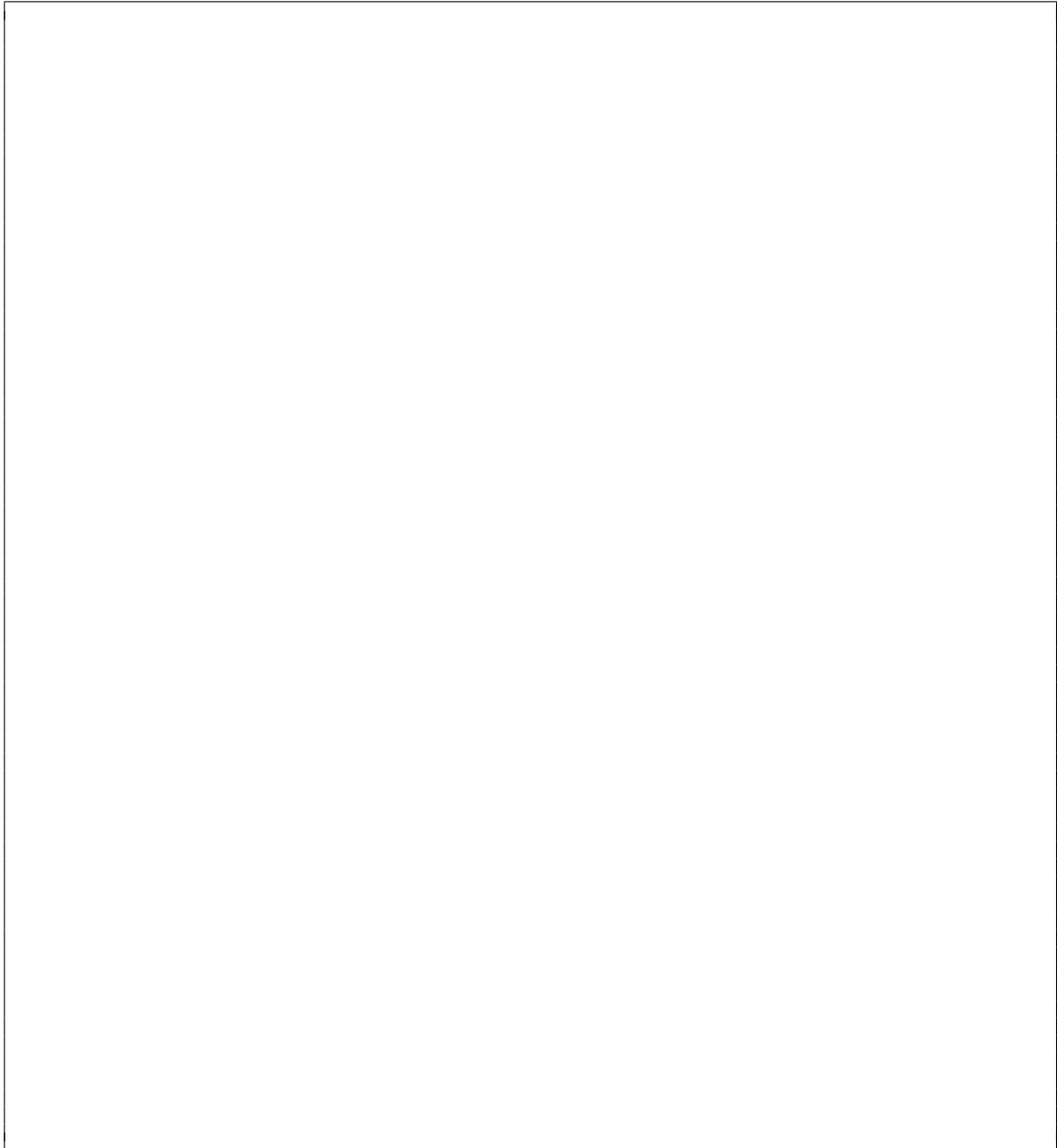
```
// doubleFilter<A>(f1: (x: A) => bool, f2: (x: A) => bool, arr: A[]): A[]  
function doubleFilter(f1, f2, arr) {  
  return arr.filter(f1).filter(f2);  
}
```

**Part a** (5 points) *Difficulty: \*\**

Provide an alternative implementation of `doubleFilter` that behaves the same as the one given above. However, it should traverse the array exactly once. Feel free to use any JavaScript feature you wish.

**Part b** (5 points) *Difficulty: \*\**

Provide an alternative implementation of `doubleFilter` that also traverses the array exactly once. However, this implementation must use `reduce` and may not use loops. (If your solution to the other part is identical to this one, that is okay.)



**End of exam.**