COMP110 – WS4 Key

1. Given the following program, write down the output of each line a-d and explain what that value represents. Refer to the slides if you need a reminder on the Predicate, Transform, and Reducer interfaces.

```
let main = async () => {
   let arr: string[] = ["Honda", "Toyota", "Jeep", "BMW", "Ford", "Fiat"];
   print(arr.filter(func).map(newType).reduce(m, 999)); // a
   print(arr.filter(torf).map(newType).reduce(comb, 0)); // b
   print(arr.filter(torf).map(newType).reduce(m, 999)); // c
   print(arr.filter(func).map(newType).reduce(comb, 0)); // d
};
let func: Predicate<string> = (str) => str.length < 5;
let torf: Predicate<string> = (str) => str.includes("a");
let newType: Transform<string, number> = (str) => str.length;
let comb: Reducer<number, number> = (memo, num) => memo + num;
let m: Reducer<number, number> = (memo, num) => {
   if (num < memo) {
       return num;
   }
   return memo;
};
main();
```

Line a: 3. The smallest number of letters out of strings whose length is less than 5.	Line b: 15. Sum of letters in strings that contain a
Line c: 4. Number of letters in the smallest string containing a	Line d: 15. Sum of letters in all strings whose length is less than 5

2. For the following questions, assume that an array of numbers called `arr` has been declared and initialized.

a. Using a while loop, print every element in `arr` that is odd and whose index is divisible by 4.	b. Using a while loop, print every third element of `arr`, starting from the end of the array (including the last element), whose index is larger than 5 (without using an if statement).

`let i = 0;`
`while (i < arr.length) {`
`  if (arr[i] % 2 === 1 && i % 4 === 0) {`
`  print(arr[i]);`
`  }`
`  i++;`
`}`
	`let i = arr.length - 1;`
	`while (i > 5) {`
	`  print(arr[i]);`
	`  i -= 3;`
	`}`

3. Given the following program, answer the questions on the right.

```
let main = async () => {
   let arr: number[] = [4, 6, 7, 2];
   let x: number = 2;
   feer(arr, x);
   print(arr); // a
   print(x); // b
};
let feer = (a: number[], x: number): boolean => {
   if (a.length >= x) {
       a[a.length] = a[a[a.length - 1] - 1];
       x = x - 1;
   }
   return true;
};
main();
```

a. What is the output at line a? (Write the elements of the array separated by commas.)

4,6,7,2,6

b. What is the output at line b?
2

4. Given the following program, answer the questions on the right.

```
let main = async () => {
   let arr = [10, 11, 12, 13, 14, 15, 16];
   print(foo(arr, (x) => x % 2 === 0 && x % 3 === 1));
};
interface FnType<T> {
   (element: T): boolean;
}
let foo = <T>(arr: T[], fn: FnType<T>) => {
   let i = 0;
   let j = 0;
   let newArr = [];
   while (i < arr.length) {
       if (fn(arr[i])) {
           newArr[j] = arr[i];
           j++;
       }
       i++;
   }
   return newArr;
};
main();
```

a. What will be the output of this program?

10,16

b. Explain what the `foo` function does.

The foo function filters the given array based on the fn given to it.

c. What is the purpose of the functional interface `FnType` (Hint: Why must it be defined)?

If this type were not defined, we would not be able to take in any predicate

5. Given the Course class below, implement the following functions.

```
class Course {
    num: number = 0; // course number
    title: string = ""; // name of the course
    professor: string = ""; // professor teaching the course
    geneds: string[] = []; // list of geneds the course fulfills
    day: string = ""; // “MWF” or “TTh”
    creditHours: number = 0; // credit hours
    difficulty: number = 0; // 1-10, where 10 is the most difficult
}
```

a. Write a **Predicate** function named `fulfillsHS` that returns `true` if a `Course`'s `geneds` array contains `"HS"`.

```
let fulfillsHS = (c: Course) => {
   let i = 0;
   while (i < c.geneds.length) {
       if (c.geneds[i] === "HS") {
           return true;
       }
       i++;
   }
   return false;
};
```

b. Write a **Reducer** function named `leastDifficultCourse` that can be used to reduce an array of `Course` objects to the `Course` with the smallest value for the `difficulty` property.

```
let leastDifficultCourse = (memo: Course, n: Course): Course => {
   if (n.difficulty < memo.difficulty) {
       return n;
   } else {
       return memo;
   }
};
```

c. Using the functions you have just written, write a function named `idealCourse` so that it's only parameter is an array of `Course` objects, and it returns the least difficult course from that array that fulfills the HS `gened`.

```
let idealCourse = (arr: Course[]): Course => {
   return arr.filter(fulfillsHS).reduce(leastDifficultCourse);
};
```