Lab 9 White Box Testing

L'obiettivo di questo laboratorio è l'esercitarsi con i test white box di piccoli moduli software.

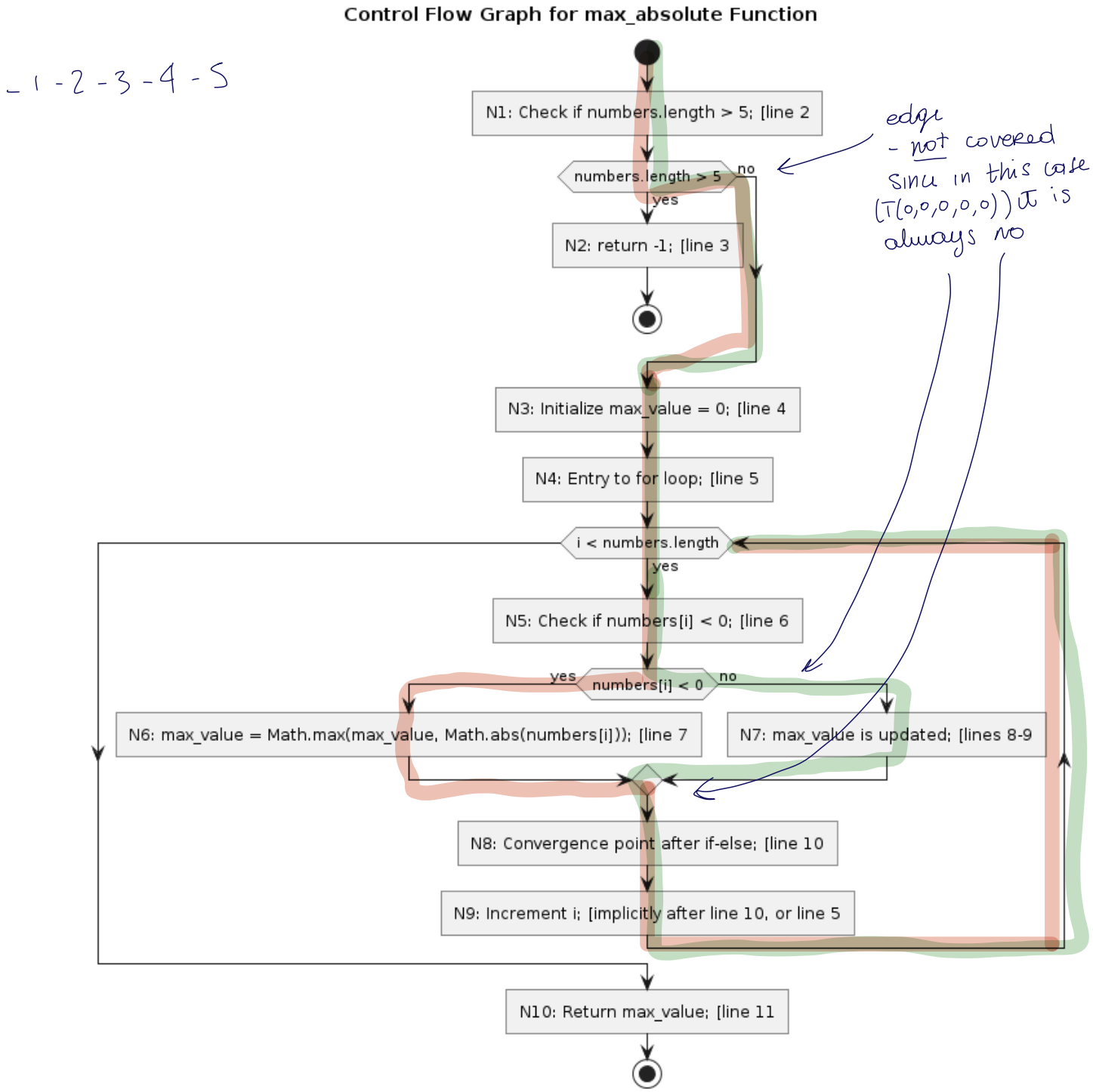
1. Definire i casi di test in un formato ad alto livello (es. tabelle), utilizzando tecniche di WB (statement coverage, decision coverage, ...)
2. In es2 e es3: confrontare i casi di test scritti usando tecniche di BB e quelli scritti usando tecniche di WB. Confrontare la coverage di WB ottenuta dai casi di test scritti usando tecniche di BB.
3. Per es4, es5, es6, scrivere i test case WB in formato ad alto livello,
4. Per es4, es5, es6, trasformare i test case scritti al punto 3 in formato codice (JS + Jest). Eseguire i casi di test e verificare la coverage calcolata da Jest.
5. Per es7 ed es8 definire i test case per statement coverage, decision coverage, multiple condition coverage e path coverage (dove possibile)

Il progetto Jest disponibile a questo link: https://git-softeng.polito.it/d023270/jest può essere usato per creare i test case in linguaggio Typescript

Esercizio 1 – Massimo Assoluto

Questa funzione deve restituire il massimo valore assoluto di un array di interi, contenente al massimo 5 elementi. Restituisce -1 in caso di errore.

```
1 function max_absolute(numbers: number[]): number {
2     if (numbers.length > 5)
3       return -1;
4     let max_value = 0;
5     for (let i = 0; i < numbers.length; i++) {
6       if (numbers[i] < 0)
7         max_value = Math.max(max_value, Math.abs(numbers[i]));
8       else
9         max_value = Math.max(max_value, numbers[i]);
10    }
11    return max_value;
12 }
```


Control Flow Graph for max_absolute Function
N1: Check if numbers.length > 5; [line 2
numbers.length > 5
no
yes
N2: return -1; [line 3
N3: Initialize max_value = 0; [line 4
N4: Entry to for loop; [line 5
i < numbers.length
yes
N5: Check if numbers[i] < 0; [line 6
yes
numbers[i] < 0
no
N6: max_value = Math.max(max_value, Math.abs(numbers[i])); [line 7
N7: max_value is updated; [lines 8-9
N8: Convergence point after if-else; [line 10
N9: Increment i; [implicitly after line 10, or line 5
N10: Return max_value; [line 11

Calcolare la coverage per i casi di test indicati

Test cases	Nodes covered	Node coverage per test case	Node coverage per test suite	Edge covered	Edge coverage per test case Per test suite	Loop in line 5
all_equals = {0,0,0,0,0};						
all_positive = {1,2,3,4,5};						
all_negative = {-1,-2,-3,-4,-5}						
[] out_of_size = {1,2,3,4,5,6};						
mixed = {-10,10,3,5,-6};						
empty = {};						
One {1}						

Esercizio 2 – Convert Int

Una funzione converte una sequenza di caratteri in un numero intero. La sequenza può iniziare con un '-' (numero negativo). Se la sequenza è più corta di 6 caratteri, viene riempita con spazi (sul lato sinistro). Il numero intero deve essere nell'intervallo minint = -32768 a maxint = 32767. La funzione segnala un errore se la sequenza di caratteri non è consentita.

```
1  class ConvertInt {
2    public convert(str: string[]): number | Error {
3     if (str.length > 6)
4       throw new Error("Invalid length");
5     let number = 0, digit, i = 0;
6     if (str[0] === '-')
7       i = 1;
8     for (; i < str.length; i++) {
9       digit = str[i].charCodeAt(0) - '0'.charCodeAt(0);
10      number = number * 10 + digit;
11    }
12    if (str[0] === '-')
13      number = -number;
14    if (number > 32767 || number < -32768)
15       throw new Error("Number out of range");
16     return number;
17   }
18 }
```

Esercizio 3 – Coda di eventi

Una coda di eventi in un sistema di simulazione riceve eventi. Ogni evento ha un tag temporale. È possibile estrarre eventi dalla coda, l'estrazione deve restituire l'evento con il tag temporale più basso. La coda scarta eventi con tag temporale negativo o nullo. La coda deve accettare almeno 100.000 eventi. Gli eventi con lo stesso tag temporale devono essere fusi (cioè il secondo ricevuto viene scartato).

```
1 class EventsQueue {
2   private queue: number[] = [];
3
4   public insert(event: number): void {
5     let index = 0;
6     while (index < this.queue.length && this.queue[index] < event) {
7       index++;
8     }
9     this.queue.splice(index, 0, event);
10  }
11
12  public pop(): number {
13    if (this.queue.length === 0) {
14       return -1;
15    }
16    return this.queue.shift()!;
17  }
18
19  public print(): void {
20    this.queue.forEach(event => console.log(event + " "));
21  }
22 }
```

Esercizio 4 – Acceptable to Eat

```
1 class Example4 {
2   static acceptableToEat(carb: number, protein: number, fat: number): boolean {
4     if (carb < 0 || protein < 0 || fat < 0) {
5       return false;
6      }
7     const totalCalories = 4 * carb + 4 * protein + 9 * fat;
8     const ratio = (carb + protein) / fat;
9     return totalCalories < 1000 && ratio > 0.5;
10   }
11 }
```

Esercizio 5 – Bike Rental (computeFee)

```
1  class Example5 {
2    static computeFee(duration: number, minRate: number, minRate2: number): number {
3      if (duration < 0 || minRate < 0 || minRate2 < 0) {
4        return -1.0;
5      }
6      if (duration < 30) {
7        return 0.0;
8      }
9      if (duration >= 30 && duration < 90) {
10       return minRate * (duration - 30);
11     }
12     return 60 * minRate + (duration - 90) * minRate2;
13   }
14  }
```

Esercizio 6 – Railway Tiket (computeFee2)

```
1  class Example6 {
2   static computeFee(basePrice: number, n_passengers: number, n_over18: number, n_under15: number): number {
3     if (basePrice < 0 || n_passengers < 0 || n_over18 < 0 || n_under15 < 0) {
4       return -1.0;
5     }
6     if (n_over18 + n_under15 > n_passengers) {
7       return -1.0;
8     }
9     if (n_passengers >= 2 && n_passengers <= 5 && n_over18 > 0) {
10     return basePrice * (n_passengers - n_under15);
11    } else {
12      return basePrice * n_passengers;
13    }
14  }
15 }
```

Esercizio 7 – Tax Calculator

Definire i test case per statement coverage, decision coverage e path coverage (se possibile).

```
1 public elonianTaxCalculator(income: number, nDependents: number): number {
2    if (income < 0) {
3      console.log("You cannot have a negative income.\n");
4      return -1;
4    }
5    if (nDependents <= 0) {
6      console.log("You must have at least one dependent.\n");
7      return -2;
8    }
9
10   let TaxSubTotal: number;
11   if (income < 10000) {
12      TaxSubTotal = 0.02 * income;
13    } else if (income < 50000) {
14      TaxSubTotal = 200 + 0.03 * (income - 10000);
15    } else {
16      TaxSubTotal = 1400 + 0.04 * (income - 50000);
17    }
18
19    let Exemption = nDependents * 50;
20    let TaxTotal = TaxSubTotal - Exemption;
21
22    if (TaxTotal < 0) {
23      TaxTotal = 0;
24    }
25
26    console.log("Elbonian Tax Collection Agency \n");
27    console.log("Tax Bill \n");
28    console.log("Citizen's Income: " + income + '\n');
29    console.log("Tax Subtotal: " + TaxSubTotal + '\n');
30    console.log("Number of Dependents: " + nDependents + '\n');
31    console.log("Tax Exemption: " + Exemption + '\n');
32    console.log("Final Tax Bill: " + TaxTotal + '\n');
33
34    return TaxTotal;
35 }
```

Esercizio 8 – foo

Definire i test case per statement coverage, decision coverage, multiple condition coverage e path coverage (se possibile).

```
1 function foo(a: number, b: number, c: number, d: number, e: number): number {
2    if (a === 0) {
3      return 0;
4    }
5    let x = 0;
6    if (a === b || (c === d && Example8.bug(a))) {
7      x = 1;
8    }
9    e = 1 / x;
10   return e;
11 }
12
13 function bug(a: number): boolean {
14   return a === 1;
15 }
```