Partie A : Question 1

Collez la capture d'écran de la sortie de la partie A.

```
PS C:\Users\coral\Documents\CodeSource\CodeSource\Partie-A> python random_fuzzer.py
trial: 0
input: "8*=(>)7;=;?;8" +%*.=/!(1:<$*8 -#)>7,(*90';
The random seed is: 2134091
Traceback (most recent call last):
  File "C:\Users\coral\Documents\CodeSource\CodeSource\Partie-A\random_fuzzer.py", line 15, in <module>
    test_script.crash_if_too_long(inp)
  File "C:\Users\coral\Documents\CodeSource\CodeSource\Partie-A\test_script.py", line 3, in crash_if_too_long
    raise ValueError
ValueError
```

Partie A : Question 2

Quel est le type de fuzzer dans la partie A ? Expliquez brièvement votre raisonnement.

Blackbox. En effet, il s'agit d'un fuzzer qui n'a aucune connaissance de l'implémentation interne de l'application testée. Il traite l'application comme une boîte noire et génère des entrées aléatoires selon le matricule.

Partie B : Question 3

Collez la capture d'écran de la sortie de la partie B.

```
PS C:\Users\coral\Documents\CodeSource\CodeSource\Partie-B> python mutation_fuzzer.py
input https ://www.polymtl.ca/
input https ://wwwnplymtl.ca/
input https:/?www%.pW/ymtlcc/
input https(:/www.pglymt.a/
input https 1?//w);.pymtl.ca/.
input ht@tps //Www.olymucl.c/
input shtutps ://wwnpolymtl.ca/
input httpgs ://w.pDly-tl.cb/
input httpgs ://w.pDly-tl.cb/
input ttsY ://w7wpodymph.ca/
input ttmps ://www]odSymtn.csa/
input htps //Www&polymtl.ca-
input httPs :+/wwH.pSoymtm.ca/
input https~ B*/gww.polymtlca/
input http )R://ww.olymt.ca/
input https ://wvw.polymtl.Eca/
input tps ://www.`olyMtl.aao
input h,tps/ww.poymPl.ca/
input httpc :/o/ww.polynmtlY.ca/
0.0 of the generated inputs are valid URLs
The random seed is: 2134091
```

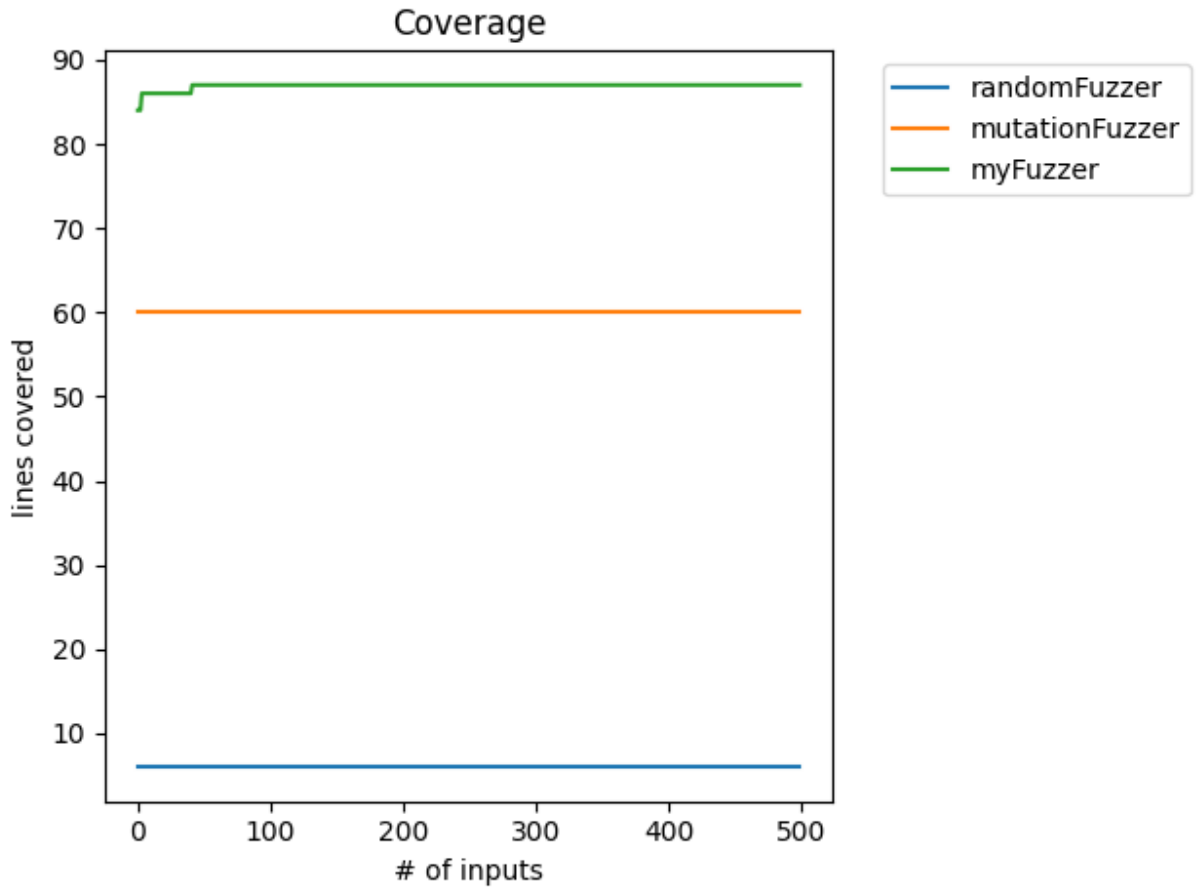
Partie B : Question 4

Quel est le type de fuzzer dans la partie B ? Expliquez brièvement votre raisonnement.

Greybox. Il s'agit d'un fuzzer qui a une certaine connaissance de l'implémentation interne de l'application testée, mais pas une connaissance complète. Il utilise cette connaissance partielle pour générer des entrées de test qui sont plus ciblées et spécifiques, avec un format url, que les entrées générées par un blackbox fuzzer, mais moins ciblées et spécifiques que celles générées par un whitebox fuzzer.

Partie C : Question 5 a)

Collez la capture d'écran du graphique que vous avez généré pour comparer les informations de couverture de Random Fuzzer, de Mutation Fuzzer et votre Fuzzer.



Partie C : Question 5 b)

On remarque que la qualité de la couverture de code est liée au type de fuzzer. En effet, myFuzzer, de type whitebox, est celui qui possède la meilleure couverture de code et est la seule qui ne reste pas constante peu importe le nombre d'entrées. De l'autre côté, randomFuzzer, de type blackbox, possède une couverture de code de moins de 10 lignes. Finalement, mutationFuzzer, de type greybox, se situe justement avec une performance qui se retrouve entre celle de la blackbox et celle de la white box.

Partie C : Question 5 c)

Incluez le code que vous avez implémenté en tant que q5.py (s'il s'agit d'un seul fichier) ou q5.zip (s'il y a plusieurs fichiers) avec votre rapport. Incluez la capture d'écran avec le code que vous avez ajouté, y compris l'implémentation du fuzzer, à votre rapport.

```
class MyFuzzer(Fuzzer):
    def __init__(self):
        self.min = -9 * 10 ** 10
        self.max = 9 * 10 ** 10

    def fuzz(self):
        return random.uniform(self.min, self.max)
```

```
def plot(cumulative_coverage, fuzzer_type):
    plt.plot(cumulative_coverage, label=fuzzer_type)
    plt.title('Coverage')
    plt.xlabel('# of inputs')
    plt.ylabel('lines covered')
    plt.legend(bbox_to_anchor=(1.05, 1.0), loc="upper left")
```

```
trials = 500
randomFuzzer = RandomFuzzer()
mutationFuzzer = MutationFuzzer(seed=["3452020"])
myFuzzer = MyFuzzer()


def fuzzer_coverage(fuzzer):
    input_set = []
    for i in range(0, trials):
        input_set.append(fuzzer.fuzz())
    return calculate_cumulative_coverage(input_set, num2words)


plot(fuzzer_coverage(randomFuzzer), "randomFuzzer")
plot(fuzzer_coverage(mutationFuzzer), "mutationFuzzer")
plot(fuzzer_coverage(myFuzzer), "myFuzzer")
plt.tight_layout()
plt.show()
```

Partie C : Question 6

Quel est le type de fuzzer de mutation dans cette partie ? Expliquez brièvement votre raisonnement.

Whitebox. Il s'agit d'un fuzzer qui a une connaissance complète de l'implémentation interne de l'application testée. Il utilise cette connaissance pour générer des entrées de test qui sont plus ciblées et spécifiques à num2words.