HOMEWORK 2

L’obiettivo del seguente homework è stato quello di raggiungere il 100% code coverage con il minor numero di test possibili.

Innanzitutto abbiamo dedicato del tempo alla corretta comprensione del metodo sui cui stiamo lavorando.

L'obiettivo del programma è contare il numero di occorrenze di una determinata parola all'interno di una stringa. Gli input sono una stringa “str” e una parola “word”. L'output è un intero che rappresenta il numero di occorrenze di “word” in “str”. Il programma prende una stringa in input, sostituisce i caratteri numerici con 'X', quindi divide la stringa in sottostringhe separate da spazi. Conta quante volte la parola “word” appare tra queste sottostringhe e restituisce il conteggio.

Successivamente abbiamo provato il programma con diversi inputs per verificare il comportamento di quest’ultimo al variare di questi. Abbiamo iniziato sugli input individualmente e successivamente abbiamo identificato i test che formeranno la suite. Testando gli input come nulli e vuoti, abbiamo notato che “word” non è in alcun modo controllata circa cosa contiene; quindi da un punto di vista del code coverage, con minor test, non ci sono test su “word” almeno per il white-box. Abbiamo quindi, inizialmente, identificato 4 casi di test su cui concentrarci :

1. Input str non vuota, con almeno una occorrenza di word.

2. Input str non vuota, senza occorrenze di word.

3. Input str vuota.

4. Input str nullo.

Avendo scritto questi test e testato il code coverage , abbiamo notato che non comprendeva le righe riguardanti la sostituzione del numero in “str”. Abbiamo quindi mantenuto sempre questi 4 test ma abbiamo aggiunto nella “str” un numero. Con questo accorgimento siamo riusciti in solo 4 test ad ottenere un 100 % code coverage. Ci teniamo a specificare che nonostante questa percentuale, i test non sono assolutamente esaustivi. Non sono infatti considerati diversi casi limite e l’assenza di controllo su “word” comporta un errore progettuale notevole.

Innanzitutto in presenza di “word” come null il programma crusha. Compilato come vuoto o spazi il programma ritorna 0 match. Ci teniamo che a specificare che nonostante “word” sia letteralmente definibile come parola, il suo campo può essere compilato come caratteri speciali, numeri e punteggiatura. In questi 3 casi il programma funziona normalmente seppur troviamo insensato compilarlo come numero nel momento in cui il programma sostituisce tutti i numeri dalla frase in cui cercare il match. Con dei controlli su “word” sarebbe possibile evitare queste ambiguità.

Altri due casi limiti sono la compilazione di “str” come solo numeri e come solo spazi. In entrambi i casi il programma si comporta in modo adeguato nella misura che non crusha e non trova match insensati (partendo dal presupposto che “word” sia diverso da null). E’ interessante notare come se assegnato null a “word”, il programma non crusha se “str” è inizializzata con spazi.

Altro limite del programma è la sua impossibilità nel distinguere una parola seguita da punteggiatura. Nell’ esempio : “Ho fame.” -> str “fame” -> word

Non ci saranno match, perché il programma considera “fame.” come un’unica parola.

Il programma non è case-sensitive.