Meno:	Filip Zubaj	Hodnotenie projektu: (max 10(TS)/5(RAM) bodov)
Cvičenie:	Štvrtok 13:00	
Dátum:	17.4.2023	

Projekt TZIV LS2022/23 – TS

Zadanie:	**17. ÚPRAVA OBRÁZKOV** Na vstupe je obrázok veľkosti nxn zapísaný po riadkoch oddelených znakom podčiarknutie a príznak S, I, H. Obrázok je čiernobiely, 1 – biela farba, 0 – čierna farba. Navrhnite Turingov stroj - riešenie, ktorý konvertuje obrázok podľa definovaného príznaku. S – obrázok bude vypísaný po stĺpcoch, I – inverzia obrázku (čierna farba na bielu a naopak), H – preklopenie obrázku horizontálne. Výstup bude nasledovať za vstupom za znakom oddeľovača $.
Vstup:	Akceptované vstupy: 1_I$; 01_10_S$; 011_111_110_H$; 1111_0000_1010_0101_I$ … (všetky vstupy nxn) Neakceptované vstupy: 201_111_111_S$; 000_001_111S$; 101H$ … vstupy s neplatnými znakmi a nesprávnym počtom podtržítiek
Neformálne riešenie:	Ako prvé kontrolujem, o akú úpravu obrázku ide, v závislosti od toho pokračujem do ďalších stavov. V prípade **I** ide o **inverziu**. V inverzii jednoducho skontrolujem prvý znak, označím J ak je 1 a zapíšem nakoniec 0 alebo N, ak je 0 a na koniec zapíšem 1. Na mieste podtržítka, ho označím pomocou U(ako underscore) a kontrolujem znak napravo od neho. Ak je napravo číslo, pokračujem ďalej, lebo ešte nie som na konci, ak tam je však I, idem do stavu *final*, čiže akceptačného stavu. Ak sa tam nachádza písmenko **S**, ideme **po stĺpcoch**. V tomto prípade si malým písmenkom najprv označím všetky znaky, ktoré budú tvoriť novú n-ticu zmeneného obrázku, a potom ich po jednom prepisujem. Robím to takto preto, aby som vždy označil rovnaký stĺpec. Najprv označujem na malé písmenko, a po zápise na veľké písmenko (J- jednotka, N - nula). Po prepise všetkých malých písmen a zápise na pravú stranu, kontrolujem, či naľavo od podtržítka je číslo alebo označenie. Ak je tam číslo, neprešli sme všetky prvky, takže zapíšem podtržítko na voľné miesto, a znovu označujem malými písmenkami. Ak je tam označenie, sú prejdené všetky znaky a môžeme ukončiť úpravu. Ak sa vyskytuje na mieste **H**, ide o **horizontálne otočenie**. V tomto prípade nájdeme prvé podtržítko pred číslami (to prvé pred H odignorujem) a od neho postupne označujeme a zapisujeme čísla (znovu J - 1, N - 0). Po zapísaní danej n-tice, sa dostaneme na U, ktorým sme označili prvé podtržítko odzadu. Ak smerom doľava od neho nájdeme podtržítko, nachádza sa tam ďalšia n-tica, a tak zapíšeme na koniec podtržítko a ideme zapisovať ďalej. Ak by naľavo od posledného U odzadu nebolo podržítko, ale voľné miesto, prešli sme všetky znaky a proces ukončujeme.

Zložitosť riešenia:	Moje riešenie vyšlo na 129 riadkov kódu a má 27 stavov. V mojom riešení je časovo najmenej náročná zmena pomocou I, čo je inverzia. Je to preto, pretože ideme postupne zľava doprava, a tak sa môžem zastavovať a otáčať na označených znakoch. To pri S napríklad nefunguje, lebo mám označený každý ntý prvok za podčiarkovníkom. Druhé najefektívnejšie je H - horizontálne preklopenie, tam sa vždy vraciam na podtržítku poprípade označenom znaku za podtržítkom, takže je to podobne zložité ako inverzia (jemne dlhšie). Najdlhším je po stĺpcoch, práve preto, lebo sa musím vždy vrátiť na úplný začiatok. Samozrejme, so zvyšujúcim vstupom sa zvyšuje aj časová náročnosť každého jedného spôsobu. To znamená, že zložitosť bude približne lineárna (O(n)). V mojom zadaní pri zložitosti nezáleží na jednotlivých znakoch, čiže či je farba čierna alebo biela, ale závisí od dĺžky vstupu. Čím dlhší vstup, tým dlhšie trvá aby sa program vykonal a zmenil obrázok. Pri funkcii inverzie, sa môj program správa nasledovne. Niektoré stavy sú konštantné, napríklad *first_check_Invert* alebo stav *start*, ktorý sa vždy vykoná iba raz. Stavy, ako *find_first_Invert*, je zavolaná n * n + n krát, pretože musí prejsť všetky čísla, ktorých je n * n + n podčiarkovníkov. *Write_zero* a *write_one* závisia od počtu núl/jednotiek na vstupe, ale maximálne sa vykonajú n * n krát, ak vo vstupe sú samé nuly/jednotky alebo 0-krát ak vo vstupe nie je žiadna jednotka poprípade nula. Dokopy sa však vykonajú n*n krát. Zápis podčiarkovníkov sa vykoná n-1 krát, lebo posledný vynechávam, aby výstup sedel so vzorom. Funkcia závisí od dĺžky vstupu, čím dlhší, tým dlhšie program beží. Na začiatok vstupu sa vraciame len raz, ďalej sa pri už prečítaných znakoch otáčame skôr. Pri funkcii H a S je to so zápisom núl a jednotiek rovnako – zapíšu sa rovnaký počet-krát. Pri H sa vždy vraciame na najbližšie neprepísane podtržítko, a tak sa na úplný začiatok vstupu dostaneme iba na konci. Funkcia H je na tom s časovou zložitosťou podobne, ako I. Najpomalšie funkcia je S. Pri nej najprv označujem a až potom znova prechádzam vstup a zapisujem čísla. Na začiatok sa vraciam omnoho častejšie ako v predošlých funkciách. Vždy pred označením, čo je n-krát a potom pri zápise každého znaku n*n krát. Vraciam sa, aj keď sú už označené znaky zapísané, pri kontrole podtržítka opäť n-krát. Dokopy to je teda n^2 + 2 * n krát. Napríklad pri 2x2 obrázku to je 8-krát.
Simulátor:	turingmachine.io

Definícia výpočtového modelu (prechodová funkcia), kód simulátora (copy-paste):

```
input: '1000_1001_1010_1111_S$'
blank: ' '
start state: start
table:
  start:
    [1,0,_]: R
    I: {L: find_first_Invert}
    H: {L: back_Horizontal}
    S: {L: back_Stlpce}

  find_first_Invert:
    [1,0,_, $, I]: L
    [' ', J, N, U]: {R: first_check_Invert}
```

```
first_check_Invert:
  1: {write: J, R: write_zero_Invert}
  0: {write: N, R: write_one_Invert}
  _: {write: U, R: check_underscore_Invert}

check_underscore_Invert:
  [1, 0, J, N]: {R: write_underscore_Invert}
  I: {R: final}

write_zero_Invert:
  [1, 0, _, $, I]: R
  ' ': {write: 0, L: find_first_Invert}

write_one_Invert:
  [1, 0, _, $, I]: R
  ' ': {write: 1, L: find_first_Invert}
write_underscore_Invert:
  [1, 0, _, $, I]: R
  ' ': {write: _, L: find_first_Invert}

back_Horizontal:
  _: {write: U, L: return_Horizontal}

get_to_middle_Horizontal:
  [1,0,_,$]: L
  H: {L: return_Horizontal}

return_Horizontal:
  [1,0,U, J, N]: L
  _: {R: writing_Horizontal}
  ' ': {R: writing_Horizontal}

writing_Horizontal:
  [J,N]: R
  1: {write: J, R: write_one_Horizontal}
  0: {write: N, R: write_zero_Horizontal}
  U: {L: find_underscore_Horizontal}

find_underscore_Horizontal:
  [J,N, U]: L
  _: {write: U, R: write_underscore_Horizontal}
  ' ': {R: final}

write_one_Horizontal:
  [1,0,U,J,N, H, $,_]: R
  ' ': {write: 1, L: get_to_middle_Horizontal}

write_zero_Horizontal:
  [1,0,U,J,N, H, $, _]: R
```

```
  ' ': {write: 0, L: get_to_middle_Horizontal}


write_underscore_Horizontal:
  [1,0,U,J,N, H, $,_]: R
  ' ': {write: _, L: get_to_middle_Horizontal}

back_Stlpce:
  [1, 0, _, j, n, J,N]: L
  ' ': {R: signing_Stlpce}

signing_Stlpce:
  [J,N]: R
  1: {write: j, R: next_Stlpce}
  0: {write: n, R: next_Stlpce}
  S: {L: back_1_Stlpce}

next_Stlpce:
  [1,0,j,n,J,N]: R
  _: {R: signing_Stlpce}

find_S_Stlpce:
  [0,1,_, $]: L
  S: {L: back_1_Stlpce}

find_S_again_Stlpce:
  [0,1,_, $]: L
  S: {L: back_Stlpce}

back_1_Stlpce:
  [1, 0, _, j, n, J, N]: L
  ' ': {R: writing_Stlpce}

writing_Stlpce:
  [J,N, 0, 1,_]: R
  j: {write: J, R: writing_one_Stlpce}
  n: {write: N, R: writing_null_Stlpce}
  S: {L: checking_Stlpce}

checking_Stlpce:
  _: L
  [1, 0]: {R: write_underscore_Stlpce}
  [J, N]: {R: final}

write_underscore_Stlpce:
  [1,0,n,N,j,J,S, $, _]: R
  ' ': {write: _, L: find_S_again_Stlpce}

writing_one_Stlpce:
  [1,0,n,N,j,J,S, $, _]: R
```

```
  ' ': {write: 1, L: find_S_Stlpce}

writing_null_Stlpce:
  [1,0,n,N,j,J,S, $, _]: R
  ' ': {write: 0, L: find_S_Stlpce}

final:
```