Due Date: Friday 29 April, 4pm

Please submit a **PDF** by email to: teaching@biods.org with subject "341 Assignment 2". Make sure that your name and ID number are on the PDF called: YourName_assignment2.pdf

What you hand in should be all your own work.

Note that **marks might be deduced** if you fail to follow the above instructions. Do let us know if you find them challenging.

Questions

1. Let $\Sigma = \{0, 1\}$, and for a word $w \in \Sigma^*$ define $\bar{w}$ to be the word obtained by exchanging 0's and 1's in w (so if $w = 101$ then $\bar{w} = 010$). Consider the language L over the alphabet $\{0, 1, 2\}$ consisting of all words of the form $w2\bar{w}$, where $w \in \Sigma^*$.

   (a) Is L regular? Prove it. (**2 marks**)

   L is not regular. We prove this by using the Pumping Lemma.

   For contradiction we assume that L is regular. Then we can apply the Pumping Lemma on the word $z = 0^k21^k \in L$. For each decomposition of $z = uvw$ with $|u| + |v| \leq k$ and $|v| > 0$ we know that both u and v only consist of 0's. The word uv^iw is in L for all $i \geq 0$, according to the Pumping lemma. However, uv^2w contains more 0's than 1's, since $|v| > 0$ and is therefore not an element of L, according to the definition of L. So we have a contradiction and can follow that L is not regular.

   (b) Is L context-free? Prove it. (**2 mark**)

   L is not context-free. We prove this by using Pumping Lemma 2.

   For contradiction we assume that L is context-free. Then we can apply Pumping Lemma 2 on the word $z = 0^k1^k21^k0^k$. We consider every possible decomposition of z into five substrings $z = uvwxy$ with $|vwx| \leq k$ and $|v| + |x| > 0$:

   i. vwx is a substring of the first part of z, only containing 0's and/or 1's. Then uv^2wx^2y, which is in L according to the Pumping Lemma, contains more characters before the 2, than after it. Therefore, uv^2wx^2y is not in L, according to the definition of L.

   ii. vwx contains the 2 in the middle of z. All other elements in vwx are 1's because $|vwx| \leq k$. If v or x contains the 2, the word uv^0wx^0z, which is in L according to the Pumping Lemma, contains no 2 any more and is therefore not in L, according to the definition of L. If on the other side neither v nor x contains the 2, the word uv^0wx^0z, which is in L according to the Pumping Lemma, contains less 1's than 0's and is therefore not in L, according to the definition of L.

   iii. vwx is a substring of the last part of z, only containing 0's and/or 1's. Then uv^2wx^2y, which is in L according to the Pumping Lemma, contains more characters before the 2, than after it. Therefore, uv^2wx^2y is not in L, according to the definition of L.

   In all cases we reach a contradiction., so our assumption was wrong. Hence, L is not context-free.

2. Design a Turing machine to compute $f(x) = 2^x$. Use comments (or self-explanatory state names, or both) in your code to make it readable. Both correctness and readability of your code will be marked. (**3 marks**)

At first the TM adds a 2 behind the input sequence of 1's to separate the input from the output. Right after that 2 a 1 is added, which is the default output (if the input is the empty string, the output is $2^0 = 1$).

Then the head moves back to the start of the input and a loop starts. In each loop at first the first 1 of the input is converted to a β. Then the head moves to the start of the output string and doubles it.

The TM doubles the output string by changing all 1's of the previous output to 0. Then it changes them back to 1's from the right to the left and appends for each of these changes a 1 at the end of the output. The doubling is complete if the TM cannot find a 0 on the right of the 2. then the head moves back to the start of the input string and the loop starts again. If there is no 1 of the input string left, which means that there is a β on the left of the 2, the TM converts the 2 to a β and accepts.

# Add 21 right behind the input string:

$q_{init}1 \rightarrow q_{init}1R$

$q_{init}\beta \rightarrow q_{defaultOutput}2L$

$q_{defaultOutput}\beta \rightarrow q_{toInputStart}1L$

# Return to the start of the input

$q_{toInputStart}1 \rightarrow q_{toInputStart}1L$

$q_{toInputStart}2 \rightarrow q_{toInputStart}2L$

$q_{toInputStart}\beta \rightarrow q_{deleteFirst1}\beta R$

# Convert the first 1 of the input to a β; if there is no 1 left: convert the separator 2 to β and accept

$q_{deleteFirst1}1 \rightarrow q_{toOutputStart}\beta R$

$q_{deleteFirst1}2 \rightarrow q_{accept}\beta$

# Move to the start of the output

$q_{toOutputStart}1 \rightarrow q_{toOutputStart}1R$

$q_{toOutputStart}2 \rightarrow q_{Output1to0}2R$

# The following can also be written as module DOUBLE as it doubles the output:

# At first all 1's in the output are converted to 0's.

$q_{Output1to0}1 \rightarrow q_{Output1to0}0R$

$q_{Output1to0}\beta \rightarrow q_{find0}\beta L$

# Find the first 0 on the left of the urrent position, convert it to a 1....

$q_{find0}1 \rightarrow q_{find0}1L$

$q_{find0}0 \rightarrow q_{add1}1R$

$q_{find0}2 \rightarrow q_{toInputStart}2L$

# ... and add a 1 at the end of the output for each such 0. Continue with this (loop to state q_{find0}).

$q_{add1}1 \rightarrow q_{add1}1R$

$q_{add1}\beta \rightarrow q_{find0}1L$

3. Show that, if there is a language Q that is both in $\mathcal{P}$ and $\mathcal{N}P$-complete, then $\mathcal{P} = \mathcal{NP}$. (**3 marks**)

According to the definition of $\mathcal{NP}$-complete, any problem in $\mathcal{NP}$ polynomially reduces to L, if L is $\mathcal{NP}$-complete. Let X be any one problem in $\mathcal{NP}$, then X polynomially reduces to L. According to the definition of the polynomial reduction, if L is in $\mathcal{P}$, then X can be solved in polynomial time. So, any problem X in $\mathcal{NP}$ is also in $\mathcal{P}$. Therefore, $\mathcal{NP} \subseteq P$, if L is in both P and $\mathcal{NP}$-complete. Furthermore, $\mathcal{P} \subseteq \mathcal{NP}$ (by definition), so $\mathcal{P} = \mathcal{NP}$.