Question 1: Playing around with Strings

Show every step of your calculations for the following questions. Please always complete your calculations; for example, in your answer write 3125 instead of 5^5.

(a) Which integer is associated with the string “santa” on the alphabet A = {a, b, c, n, s, t}? In our notation, “santa” becomes s\_5s\_4s\_3s\_1. The integer x associated with that string is:

\[ x = 5 \cdot 6^4 + 1 \cdot 6^3 + 6 \cdot 6^1 + 1 \cdot 6^0 = 6480 + 216 + 36 + 1 = 6877. \]

(b) What is the string on the alphabet B = {a, d, e, h, k, l, s, w} that is associated with the integer 1112822?

Here, n = |B| = 8, so we have:

\[
\begin{align*}
1112822 / 8 & = 139102 \text{ R } 6 \\
139102 / 8 & = 17387 \text{ R } 6 \\
17387 / 8 & = 2173 \text{ R } 3 \\
2173 / 8 & = 271 \text{ R } 5 \\
271 / 8 & = 33 \text{ R } 7 \\
33 / 8 & = 4 \text{ R } 1 \\
4 / 8 & = 0 \text{ R } 4
\end{align*}
\]

Therefore, we get the string “haskell.”

(c) Let n be a natural number and w\_A and w\_B be the strings associated with n on the alphabets A and B, respectively, where |A| = 3 and |B| = 4. Give an example of a number n for which the length of w\_A is 4 and the length of w\_B is 3.

One way to approach this is to compute the smallest number whose base 3 representation has 4 symbols. That is s\_1s\_2s\_3s\_1, associated with the integer n = 27 + 9 + 3 + 1 = 40. Let us now determine the base 4 representation of n:
Therefore, the base 4 representation is $s_2s_1s_3$. As expected, its length is 3, and so $n = 40$ is a solution of our problem.

(d) Compute $\text{UPCHANGE}_{3,5}(100)$.

Base 3 representation of 100 is:

\[
\begin{align*}
100 / 3 &= 33 \text{ R } 1 \\
33 / 3 &= 10 \text{ R } 3 \\
10 / 3 &= 3 \text{ R } 1 \\
3 / 3 &= 0 \text{ R } 3
\end{align*}
\]

We get $w = s_3s_1s_3$. In base 5 notation, this is:

\[
x = 3 \cdot 5^3 + 1 \cdot 5^2 + 3 \cdot 5^1 + 1 \cdot 5^0 = 375 + 25 + 15 + 1 = 416. \text{ This is the solution.}
\]

(e) Compute $\text{DOWNCHANGE}_{7,9}(201)$.

Base 9 representation of 201 is:

\[
\begin{align*}
201 / 9 &= 22 \text{ R } 3 \\
22 / 9 &= 2 \text{ R } 4 \\
2 / 9 &= 0 \text{ R } 2
\end{align*}
\]

We get $w = s_2s_4s_3$. In base 7 notation, this is:

\[
x = 2 \cdot 7^2 + 4 \cdot 7^1 + 3 \cdot 7^0 = 98 + 28 + 3 = 129. \text{ This is the solution.}
\]

(f) Compute $\text{LTEND}_4(37)$.

37 in base 4 representation is:

\[
\begin{align*}
37 / 4 &= 9 \text{ R } 1 \\
9 / 4 &= 2 \text{ R } 1 \\
2 / 4 &= 0 \text{ R } 2
\end{align*}
\]

We get $w = s_2s_1s_1$. Therefore, the value of the leftmost digit is 2. This is the solution.

(g) Given the alphabet $C = \{x, y, z\}$, what is the least integer whose associated string on $C$ has a length of four?

The string associated with that number must be xxxx. This is the same as $s_1s_1s_1s_1$. In base 3 notation, it has the value $n = 27 + 9 + 3 + 1 = 40$. That is the solution.
Question 2: Ordering with Post-Turing

(a) Write a Post-Turing program (you can use macros) using an alphabet $A = \{a, b\}$ that computes the following function $f$:

$$f(x, y) = \begin{cases} 
a, & \text{if } x < y \\
b, & \text{otherwise} 
\end{cases}$$

Here, $x < y$ means that the string $x$ precedes the string $y$ in alphabetical order. For example, it is true that $abba < ba$, $aaa < aab$, and $bb < bbb$.

You get bonus points if you write a program that computes $f$ strictly.

Well, it turns out that a strict computation is a really difficult task. Here is a program that achieves it:

```
RIGHT
IF B GOTO D1
[C1] RIGHT               // go to end of x1
    IF a GOTO C1
    IF b GOTO C1
    RIGHT
    IF B GOTO C4 // x2 is empty
    LEFT
[C2] LEFT                // go back to start of x1
    IF a GOTO C2
    IF b GOTO C2
[D7] RIGHT
    IF a GOTO C3
    PRINT B // erase leftmost symbol of x1
    RIGHT
    IF B GOTO B2 // next symbol in x1 is b, and it’s the last one
    GOTO B1 // next symbol in x1 is b, and there are more following
[C3] PRINT B // erase leftmost symbol of x1
    RIGHT
    IF B GOTO A2 // next symbol in x1 is a, and it’s the last one
    GOTO A1 // next symbol in x1 is a, and there are more following
[D1] PRINT B // if x1 is empty, erase x2 …
    RIGHT
    IF a GOTO D1
    IF b GOTO D1
    PRINT a // … write output “a” and terminate.
    LEFT
    GOTO E
[C4] LEFT                 // if x2 is empty and x1 is not, erase x1 …
[C5] PRINT B
    LEFT
    IF a GOTO C5
    IF b GOTO C5
    PRINT b // … write output “b” and terminate.
    LEFT
```
GOTO E

[A]  RIGHT    // move past $x_1$
         IF a GOTO A
         IF b GOTO A

[A1]  RIGHT    // find first symbol of $x_2$
         IF B GOTO A1
         IF b GOTO D12 // output “a”
         GOTO D4     // compare next pair of letters

[A2]  RIGHT    // move past $x_1$
         IF a GOTO A2
         IF b GOTO A2

[A21] RIGHT    // find first symbol of $x_2$
         IF B GOTO A21
         IF b GOTO D22 // output “a”
          GOTO D23    // output “b” (as there are no further symbols in $x_2$)

[B]   RIGHT    // move past $x_1$
         IF a GOTO B
         IF b GOTO B

[B1]  RIGHT    // find first symbol of $x_2$
         IF B GOTO B1
         IF a GOTO D13 // output “b”
         GOTO D4     // compare next pair of letters

[B2]  RIGHT    // move past $x_1$
         IF a GOTO B2
         IF b GOTO B2

[B21] RIGHT    // find first symbol of $x_2$
         IF B GOTO B21
         IF a GOTO D23 // output “b”
          RIGHT     // if the strings are equal, output “b”
         IF B GOTO D23 // output “a” (as there are additional symbols in $x_2$)

[D]   PRINT B  // delete $x_2$
         RIGHT
         PRINT B

[D1]   PRINT B
         RIGHT
         IF a GOTO D1
         IF b GOTO D1

[D11]  LEFT
         IF B GOTO D11

[D12]  PRINT B // delete $x_1$
         LEFT
         IF a GOTO D12
         IF b GOTO D12
         PRINT a   // print output
         LEFT
         GOTO E

[D13]  LEFT    // delete $x_2$
         PRINT B

[D130] PRINT B
It is much easier to perform a non-strict computation and replace the symbols that have already been processed with markers. Let us use the marker M for the first input and N for the second one. Then the following program computes f non-strictly:
[B₁] RIGHT
  IF B GOTO D₁
  IF a GOTO A₁
  PRINT M

[C₁] RIGHT  // go to end of x₁
  IF a GOTO C₁
  IF b GOTO C₁
  RIGHT
  IF B GOTO D₂  // x₂ is empty, so output “b”
  IF a GOTO D₂  // x₂ < x₁, so output “b”

[C₂] PRINT N  // print marker

[C₃] LEFT  // find rightmost M marker
  IF a GOTO C₃
  IF b GOTO C₃
  IF B GOTO C₃
  IF N GOTO C₃
  GOTO B₁  // start next cycle

[A₁] PRINT M

[C₄] RIGHT  // go to end of x₁
  IF a GOTO C₄
  IF b GOTO C₄
  RIGHT
  IF B GOTO D₂  // x₂ is empty, so output “b”
  IF b GOTO D₁  // x₁ < x₂, so output “a”
  GOTO C₂  // print marker, start next cycle

[D₁] PRINT B  // erase x₂
  RIGHT
  IF a GOTO D₁
  IF b GOTO D₁

[D₃] PRINT B  // erase x₁ (i.e., until marker M found)
  LEFT
  IF a GOTO D₃
  IF b GOTO D₃
  IF B GOTO D₃
  IF N GOTO D₃
  PRINT a  // print output
  GOTO E

[D₂] PRINT B  // erase x₂
  RIGHT
  IF a GOTO D₂
  IF b GOTO D₂

[D₄] PRINT B  // erase x₁ (i.e., until marker M found)
  LEFT
  IF a GOTO D₄
  IF b GOTO D₄
  IF B GOTO D₄
  IF N GOTO D₄
  PRINT b  // print output
(b) Write down the list of successive tape configurations that your program generates during the computation of \(f(aba, abba)\).

For the strict computation:

\[
\begin{align*}
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BabaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\text{BBbaBabbaB} \\
\text{↑} \\
\end{align*}
\]
For the non-strict computation:

BabaBabbaB
↑
BabaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑
BMbaBabbaB
↑