이재호 풀이.

문제 1

문자열 인자를 받아서 각 글자마다 \fbox를 치는 명령 \foobox를 작성하여라.

풀이 1. 처음의 \tl_head와 \tl_tail을 사용해 구현하려는 시도를 완성한 방법입니다. 매번 함수가 토큰열 전체를 읽은 후 한 글자씩 '떼어내어' 처리하는 방식입니다.

```
\ExplSyntaxOn
\cs_new:Npn \foo_fn_rec:x #1
  {
    \tl_set:Nx \l_tmpa_tl { #1 }
    \tl_if_empty:oF { \l_tmpa_tl }
      {
        \fbox { \tl_head:N \l_tmpa_tl }
        \foo_fn_rec:x { \tl_tail:N \l_tmpa_tl }
      }
  }

\NewDocumentCommand \fooboxrec { m }
  {
    \foo_fn_rec:x { #1 }
  }
\ExplSyntaxOff
```

입력: \fooboxrec{expl3}
출력: e x p l 3

풀이 2. 이번 수업에서 배운 map을 이용하여 구현한 방법입니다.

```
\ExplSyntaxOn
\cs_new:Npn \foo_fn_map:n #1
  {
    \tl_map_inline:nn { #1 }
      {
        \fbox { ##1 }
      }
  }

\NewDocumentCommand \fooboxmap { m }
  {
    \foo_fn_map:n { #1 }
  }
\ExplSyntaxOff
```

입력: \fooboxmap{expl3}
출력: e x p l 3

풀이 3. 첫 시도와는 다르게 함수가 항상 토큰열의 한 문자씩 처리하지만, 특정 중단 지점을 만나면 함수가 자기자신을 다시 부르는 것을 중단하는 방식입니다. 이번 수업에서 '재미삼아' 알려주신 방법을 사용했습니다.

```
\ExplSyntaxOn
\cs_new:Npn \foo_fn_recq:n #1
  {
    \quark_if_recursion_tail_stop:n { #1 }
    \fbox { #1 }
    \foo_fn_recq:n
  }

\NewDocumentCommand \fooboxrecq { m }
  {
    \foo_fn_recq:n #1 \q_recursion_tail \q_recursion_stop
  }
\ExplSyntaxOff
```

입력: \fooboxrecq{expl3}
출력: e x p l 3

연습문제 1. 기본 1

문제에서 제시한 \foobox에서 각 글자에 아래 예시와 같이 색상을 입혀보아라. 각 글자 사이에 1pt의 간격을 둔다.

풀이 1.

```
\ExplSyntaxOn
\cs_new:Npn \colorfoo_fn_rec:x #1
  {
    \tl_set:Nx \l_tmpa_tl { #1 }
    \tl_if_empty:oF { \l_tmpa_tl }
      {
        \fcolorbox{red}{red}{\color{yellow} \tl_head:N \l_tmpa_tl }
        \tl_set:Nx \l_tmpb_tl { \tl_tail:N \l_tmpa_tl }

        \tl_if_empty:oF { \l_tmpb_tl }
          {
            \hspace{1pt}
            \colorfoo_fn_rec:x { \tl_tail:N \l_tmpa_tl }
          }
      }
  }
```

```
\NewDocumentCommand \colorfooboxrec { m }
  {
    \colorfoo_fn_rec:x { #1 }
  }
\ExplSyntaxOff
```

입력: \colorfooboxrec{expl3}
출력: e x p l 3

풀이 2.

```
\ExplSyntaxOn
\cs_new:Npn \colorfoo_fn_map:n #1
  {
    \tl_map_inline:nn { #1 }
      {
        \fcolorbox{red}{red}{\color{yellow} ##1 }
        \hspace{1pt}
      }
      \hspace{-1pt}
  }

\NewDocumentCommand \colorfooboxmap { m }
  {
    \colorfoo_fn_map:n { #1 }
  }
\ExplSyntaxOff
```

입력: \colorfooboxmap{expl3}
출력: e x p l 3

풀이 3.

```
\ExplSyntaxOn
\cs_new:Npn \colorfoo_fn_recq:n #1
  {
    \quark_if_recursion_tail_stop_do:nn { #1 }
      {
        \hspace{-1pt}
      }
    \fcolorbox{red}{red}{\color{yellow} #1 }
    \hspace{1pt}
    \colorfoo_fn_recq:n
  }

\NewDocumentCommand \colorfooboxrecq { m }
  {
    \colorfoo_fn_recq:n #1 \q_recursion_tail \q_recursion_stop
```

```
  }
\ExplSyntaxOff
```

입력: \colorfooboxrecq{expl3}
출력: e x p l 3

연습문제 1. 기본 2

\foobox에서 인자로 주어진 글자 수를 세어서 마지막에 괄호와 함께 표현하는 명령 \barbox를 작성하여라.

풀이 1.

```
\ExplSyntaxOn
\cs_new:Npn \bar_fn_rec:xn #1 #2
  {
    \tl_set:Nx \l_tmpa_tl { #1 }
    \int_set:Nn \l_tmpa_int { #2 }
    \tl_if_empty:oF { \l_tmpa_tl }
      {
        \fcolorbox{red}{red}{\color{yellow} \tl_head:N \l_tmpa_tl }
        \tl_set:Nx \l_tmpb_tl { \tl_tail:N \l_tmpa_tl }

        \tl_if_empty:oTF { \l_tmpb_tl }
          {
            {~}( \int_use:N \l_tmpa_int )
          }
          {
            \hspace{1pt}
            \int_incr:N \l_tmpa_int
            \bar_fn_rec:xn { \l_tmpb_tl } { \l_tmpa_int }
          }
      }
  }

\NewDocumentCommand \barboxrec { m }
  {
    \bar_fn_rec:xn { #1 } { 1 }
  }
\ExplSyntaxOff
```

입력: \barboxrec{expl3}
출력: e x p l 3 (5)

풀이 2.

```
\ExplSyntaxOn
```

```
\cs_new:Npn \bar_fn_map:n #1
  {
    \int_zero:N \l_tmpa_int
    \tl_map_inline:nn { #1 }
      {
        \int_incr:N \l_tmpa_int
        \fcolorbox{red}{red}{\color{yellow} ##1 }
        \hspace{1pt}
      }
      \hspace{-1pt}
      {~}( \int_use:N \l_tmpa_int )
  }

\NewDocumentCommand \barboxmap { m }
  {
    \bar_fn_map:n { #1 }
  }
\ExplSyntaxOff
```

입력: `\barboxmap{expl3}`
출력: e x p l 3 (5)

풀이 3.

```
\ExplSyntaxOn
\cs_new:Npn \bar_fn_recq:n #1
  {
    \quark_if_recursion_tail_stop_do:nn { #1 }
      {
        \hspace{-1pt}
        {~}( \int_use:N \l_tmpa_int )
      }
    \int_incr:N \l_tmpa_int
    \fcolorbox{red}{red}{\color{yellow} #1 }
    \hspace{1pt}
    \bar_fn_recq:n
  }

\NewDocumentCommand \barboxrecq { m }
  {
    \int_zero:N \l_tmpa_int
    \bar_fn_recq:n #1 \q_recursion_tail \q_recursion_stop
  }
\ExplSyntaxOff
```

입력: `\barboxrecq{expl3}`
출력: e x p l 3 (5)

연습문제 1. 발전 3

\foobox에서 인자로 주어진 글자 수를 세어서 마지막에 괄호와 함께 표현하는 명령 \barbox를 작성하여라.

풀이 1.

```
\ExplSyntaxOn
\cs_new:Npn \oddbar_fn_rec:xn #1 #2
  {
    \tl_set:Nx \l_tmpa_tl { #1 }
    \int_set:Nn \l_tmpa_int { #2 }
    \tl_if_empty:oF { \l_tmpa_tl }
      {
        \int_if_odd:nTF \l_tmpa_int
        {
          \fcolorbox{red}{red}{\color{yellow} \tl_head:N \l_tmpa_tl }
        }
        {
          \tl_head:N \l_tmpa_tl
        }
        \tl_set:Nx \l_tmpb_tl { \tl_tail:N \l_tmpa_tl }

        \tl_if_empty:oTF { \l_tmpb_tl }
          {
            {~}( \int_use:N \l_tmpa_int )
          }
          {
            \hspace{1pt}
            \int_incr:N \l_tmpa_int
            \oddbar_fn_rec:xn { \l_tmpb_tl } { \l_tmpa_int }
          }
      }
  }

\NewDocumentCommand \oddbarboxrec { m }
  {
    \oddbar_fn_rec:xn { #1 } { 1 }
  }
\ExplSyntaxOff
```

입력: \oddbarboxrec{expl3}
출력: e x p l 3 (5)

풀이 2.

```
\ExplSyntaxOn
\cs_new:Npn \oddbar_fn_map:n #1
  {
    \int_zero:N \l_tmpa_int
```

```
    \tl_map_inline:nn { #1 }
      {
        \int_incr:N \l_tmpa_int
        \int_if_odd:nTF \l_tmpa_int
        {
          \fcolorbox{red}{red}{\color{yellow} ##1 }
        }
        {
          ##1
        }
        \hspace{1pt}
      }
      \hspace{-1pt}
      {~}( \int_use:N \l_tmpa_int )
  }

\NewDocumentCommand \oddbarboxmap { m }
  {
    \oddbar_fn_map:n { #1 }
  }
\ExplSyntaxOff
```

입력: `\oddbarboxmap{expl3}`
출력: e x p l 3 (5)

풀이 3.

```
\ExplSyntaxOn
\cs_new:Npn \oddbar_fn_recq:n #1
  {
    \quark_if_recursion_tail_stop_do:nn { #1 }
      {
        \hspace{-1pt}
        {~}( \int_use:N \l_tmpa_int )
      }
    \int_incr:N \l_tmpa_int
    \int_if_odd:nTF \l_tmpa_int
    {
      \fcolorbox{red}{red}{\color{yellow} #1 }
    }
    {
      #1
    }
    \hspace{1pt}
    \oddbar_fn_recq:n
  }

\NewDocumentCommand \oddbarboxrecq { m }
  {
```

```
    \int_zero:N \l_tmpa_int
    \oddbar_fn_recq:n #1 \q_recursion_tail \q_recursion_stop
  }
\ExplSyntaxOff
```

입력: \oddbarboxrecq{expl3}
출력: e x p l 3 (5)

문제 2

주어지는 문자열을 3개 단위로 끊어서 다음 출력예와 같이 배열하여라. 마지막 항목은 3개가 되지 않을 수 있으며 스페이스는 무시한다.

재귀를 이용한 방법입니다.

```
\ExplSyntaxOn
\tl_new:N \l_tablines_tl
\cs_new:Npn \scmd_fn_rec:n #1
  {
    \quark_if_recursion_tail_stop:n { #1 }

    \int_incr:N \l_tmpa_int
    \tl_set:Nn \l_tmpa_tl { #1 }
    \int_compare:nT { \int_mod:nn \l_tmpa_int 3 == 0 }
      {
        \int_compare:nTF { \int_mod:nn \l_tmpa_int 6 == 0 }
          {
            \tl_put_right:Nn \l_tmpa_tl { \tabularnewline }
          }
          {
            \tl_put_right:Nn \l_tmpa_tl { \c_alignment_token }
          }
      }
    \tl_put_right:Nx \l_tablines_tl { \l_tmpa_tl }
    \scmd_fn_rec:n
  }

\NewDocumentCommand \scmdrec { m }
  {
    \tl_clear:N \l_tablines_tl
    \int_zero:N \l_tmpa_int
    \scmd_fn_rec:n #1 \q_recursion_tail \q_recursion_stop
    \begin{tabular}[t]{ll}
      \l_tablines_tl
    \end{tabular}
  }
\ExplSyntaxOff
```

```
입력: \scmdrec{abcdefg hijklmn}
출력:  abc   def
       ghi   jkl
       mn
```

연습문제 2. 기본 1

명령 \acmd는 두 개의 인자를 받는다. 첫 번째 인자는 임의의 문자열이다. 만약 문자열이 지정된 숫자보다 크다면 앞에서부터 숫자에 해당되는 번째 문자까지만 출력하라. 만약 문자열이 지정된 숫자보다 작다면 문자열의 앞쪽에 _(언더스코어)를 붙여 n개의 문자열이 되도록 하라.

_를 모자란 개수만큼 출력하기 위하여 \int_while_do:nn을 사용하였습니다.

```
\ExplSyntaxOn
\int_new:N \l_count_int
\cs_new:Npn \acmd_fn:nn #1 #2
  {
    \int_set:Nn \l_tmpa_int { #1 }
    \tl_set:Nn \l_tmpa_tl { #2 }
    \int_set:Nn \l_tmpb_int
      {
        \tl_count:N \l_tmpa_tl
      }
    \int_compare:nTF { \l_tmpa_int <= \l_tmpb_int }
      {
        \int_zero:N \l_count_int
        \tl_map_inline:Nn \l_tmpa_tl
          {
            \int_compare:nT { \l_count_int < \l_tmpa_int }
              {
                ##1
              }
            \int_incr:N \l_count_int
          }
      }
      {
        \int_zero:N \l_count_int
        \int_while_do:nn { \l_count_int < \l_tmpa_int - \l_tmpb_int }
          {
            _
            \int_incr:N \l_count_int
          }
        \l_tmpa_tl
      }
  }

\NewDocumentCommand \acmd { m m }
  {
    \acmd_fn:nn { #1 } { #2 }
```

```
  }
\ExplSyntaxOff
```

입력: `\acmd{5}{beautiful}\quad \acmd{5}{abc}`
출력: beaut __abc

연습문제 2. 발전 2

Python에는 문자열을 자르는(슬라이싱) 재미있는 기법이 있다. `\myslicing` 명령을 정의하되, 3개의 인자를 받아들이도록 하여 첫 인자로 주어지는 문자열을 #2부터 #3까지 슬라이싱하여 (즉 `mystring[`m`:`n`]`과 비슷) 출력하도록 하여라. 스페이스는 무시한다.

```
\ExplSyntaxOn
\cs_new:Npn \slicing_fn:nnn #1 #2 #3
  {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \int_zero:N \l_tmpa_int
    \tl_map_inline:Nn \l_tmpa_tl
      {
        \int_incr:N \l_tmpa_int
        \int_compare:nT { #2 <= \l_tmpa_int <= #3 }
          {
            ##1
          }
      }
  }

\NewDocumentCommand \myslicing { m m m }
  {
    \slicing_fn:nnn { #1 } { #2 } { #3 }
  }
\ExplSyntaxOff
```

입력: `\myslicing{Hello world}{3}{7}`
출력: llowo

연습문제 2. 발전 3

두 번째 혹은 세 번째 인자의 크기가 주어진 첫 번째 인자의 길이보다 크다면, 마지막 값으로 처리합니다. 두 번째 혹은 세 번째 인자가 양의 정수가 아닌 경우는 고려하지 않았습니다.

```
\ExplSyntaxOn
\int_new:N \l_first_int
\int_new:N \l_second_int
\cs_new:Npn \item_swap:nnnN #1 #2 #3 #4
```

```
{
  \tl_new:N #4
  \int_set:Nn \l_first_int { #2 }
  \int_set:Nn \l_second_int { #3 }
  \int_set:Nn \l_tmpa_int { \tl_count:n { #1 } }
  \int_compare:nT { \l_first_int > \l_tmpa_int }
    {
      \int_set:Nn \l_first_int { \l_tmpa_int }
    }
  \int_compare:nT { \l_second_int > \l_tmpa_int }
    {
      \int_set:Nn \l_second_int { \l_tmpa_int }
    }

  \int_zero:N \l_tmpa_int
  \tl_map_inline:nn { #1 }
    {
      \int_incr:N \l_tmpa_int
      \int_compare:nTF { \l_tmpa_int == \l_first_int }
        {
          \tl_set:Nn \l_tmpa_tl { ##1 }
        }
        {
          \int_compare:nT { \l_tmpa_int == \l_second_int }
            {
              \tl_set:Nn \l_tmpb_tl { ##1 }
            }
        }
    }


  \int_zero:N \l_tmpa_int
  \tl_map_inline:nn { #1 }
    {
      \int_incr:N \l_tmpa_int
      \int_compare:nTF { \l_tmpa_int == \l_first_int }
        {
          \tl_put_right:Nn #4 { \l_tmpb_tl }
        }
        {
          \int_compare:nTF { \l_tmpa_int == \l_second_int }
            {
              \tl_put_right:Nn #4 { \l_tmpa_tl }
            }
            {
              \tl_put_right:Nn #4 { ##1 }
            }
        }
    }
}
```

```
\NewDocumentCommand \myitemswap { m m m m }
  {
    \item_swap:nnnN { #1 } { #2 } { #3 } { #4 }
  }
\ExplSyntaxOff
```

입력: `\myitemswap{abcde}{2}{4}{\myresult}\myresult`
출력: adcbe

문제 3

다음 실행의 결과가 어떠할지 예측해보아라. 실제로 예상과 같은지 확인해보아라.

```
\ExplSyntaxOn
\tl_new:N \l_tmpc_tl

\tl_set:Nn \l_tmpa_tl { foo }
\tl_set:Nn \l_tmpb_tl { \l_tmpa_tl }
\tl_set:No \l_tmpc_tl { \l_tmpa_tl }

\tl_set:Nn \l_tmpa_tl { bar }

\tl_use:N \l_tmpb_tl
\tl_use:N \l_tmpc_tl
\ExplSyntaxOff
```

출력: barfoo

연습문제 3. 기본 1

주어지는 문자열을 앞에서부터 3개째마다 쉼표를 추가하는 명령을 작성하여라.

```
\ExplSyntaxOn
\cs_new:Npn \test_fn:n #1
  {
    \int_zero:N \l_tmpa_int
    \int_set:Nn \l_tmpb_int
      {
        \tl_count:n { #1 }
      }

    \tl_map_inline:nn { #1 }
      {
        \int_incr:N \l_tmpa_int
        \int_compare:nTF { \int_mod:nn \l_tmpa_int 3 == 0 }
          {
```

```
            \int_compare:nTF { \l_tmpa_int == \l_tmpb_int }
              {
                ##1
              }
              {
                ##1,~
              }
          }
          {
            ##1
          }
      }
  }

\NewDocumentCommand \test { m }
  {
    \test_fn:n { #1 }
  }
\ExplSyntaxOff
```

입력: `\test{this is just a test}`
출력: thi, sis, jus, tat, est

문제 4

새로운 명령 newcmd는 다음과 같은 형식으로 실행한다. `\newcmd{abc, de, fgh, i, jkl}` 쉼표로 분리된 각 항목을 enumerate으로 배열하여라.

KTUG QnA:236820에 Zeta 필명으로 답변 작성하였습니다.

풀이 1.

```
\ExplSyntaxOn
\NewDocumentCommand \newcmdmap { m }
  {
    \begin{enumerate}
      \clist_set:Nn \l_tmpa_clist { #1 }
      \clist_map_inline:Nn \l_tmpa_clist { \item ##1 }
    \end{enumerate}
  }
\ExplSyntaxOff
```

입력: \newcmdmap{abc, de, fgh, i, jkl}
출력:

1. abc
2. de
3. fgh
4. i
5. jkl

풀이 2.

```
\ExplSyntaxOn
\cs_new:Npn \newcmd_rec:n #1
  {
    \quark_if_recursion_tail_stop:n { #1 }
    \tl_if_eq:nnTF { #1 } { , } { \item } { #1 }
    \newcmd_rec:n
  }
\ExplSyntaxOff

\NewDocumentCommand \newcmdrec { m }
  {
    \begin{enumerate}
      \item
      \newcmd_rec:n #1 \q_recursion_tail \q_recursion_stop
    \end{enumerate}
  }
```

입력: \newcmdrec{abc, de, fgh, i, jkl}
출력:

1. abc
2. de
3. fgh
4. i
5. jkl

덤으로 plainTEX을 사용한 풀이 3.

```
\def\newcmdplain#1{
  \begin{enumerate}
    \item\expandafter\newcmda#1\end%
  \end{enumerate}}
```

```
\def\newcmda#1{\ifx#1\end\let\next\relax
  \else\ifx#1,\item\let\next\newcmda
  \else#1\let\next\newcmda\fi\fi\next}
```

입력: `\newcmdplain{abc, de, fgh, i, jkl}`
출력:

1. abc
2. de
3. fgh
4. i
5. jkl

연습문제 4. 기본 1

새로운 명령 `\newcmd`는 다음과 같은 형식으로 실행한다. `\newcmd{this is just a test}` 주어지는 인자를 먼저 세 개마다 쉼표를 붙여 분리하고, 분리된 각 단어를 `resi`, `resii`, `resiii`, `resiv`, ...에 넣어 반환하라.

```
\ExplSyntaxOn
\cs_new:Npn \newcmd_fn:n #1
  {
    \tl_clear:N \l_tmpa_tl
    \int_zero:N \l_tmpa_int
    \int_set:Nn \l_tmpb_int
      {
        \tl_count:n { #1 }
      }

    \tl_map_inline:nn { #1 }
      {
        \int_incr:N \l_tmpa_int
        \int_compare:nTF { \int_mod:nn \l_tmpa_int 3 == 0 }
          {
            \int_compare:nTF { \l_tmpa_int == \l_tmpb_int }
              {
                \tl_put_right:Nn \l_tmpa_tl { ##1 }
              }
              {
                \tl_put_right:Nn \l_tmpa_tl { ##1,~ }
              }
          }
          {
            \tl_put_right:Nn \l_tmpa_tl { ##1 }
          }
```

```
      }

    \clist_set:Nx \l_tmpa_clist { \l_tmpa_tl }
    \int_zero:N \l_tmpa_int
    \clist_map_inline:Nn \l_tmpa_clist
      {
        \int_incr:N \l_tmpa_int
        \tl_set:cn { res \int_to_roman:n { \l_tmpa_int } } { ##1 }
      }
  }

\NewDocumentCommand \newcmd { m }
  {
    \newcmd_fn:n { #1 }
  }
\ExplSyntaxOff
```

입력: `\newcmd{this is just a test}\resi, \resiv`

출력: thi, tat

연습문제 4. 발전 2

인자로 주어지는 단어의 각 문자가 몇 번씩 사용되었는지를 예시와 같이 출력하여라.

```
\ExplSyntaxOn
\cs_new:Npn \testcmd_fn:n #1
  {
    \int_zero:N \l_tmpa_int
    \int_set:Nn \l_tmpb_int { \tl_count:n { #1 } }
    \tl_clear:N \l_tmpa_tl
    \tl_map_inline:nn { #1 }
      {
        \int_incr:N \l_tmpa_int
        \int_compare:nTF { \l_tmpa_int < \l_tmpb_int }
          {
            \tl_put_right:Nn \l_tmpa_tl { ##1, }
          }
          {
            \tl_put_right:Nn \l_tmpa_tl { ##1 }
          }
      }

    \clist_set:Nx \l_tmpa_clist { \l_tmpa_tl }
    \clist_sort:Nn \l_tmpa_clist
      {
        \int_compare:nTF { \pdftex_strcmp:D { ##1 } { ##2 } > 0 }
          {
            \sort_return_swapped:
          }
```

```
          {
            \sort_return_same:
          }
      }

    \tl_clear:N \l_tmpa_tl
    \int_zero:N \l_tmpa_int
    \int_zero:N \l_tmpb_int
    \clist_map_inline:Nn \l_tmpa_clist
      {
        \int_incr:N \l_tmpa_int
        \int_incr:N \l_tmpb_int
        \int_compare:nT { \pdftex_strcmp:D { ##1 } { \l_tmpa_tl } > 0 }
          {
            \tl_if_empty:NF \l_tmpa_tl
              {
                \l_tmpa_tl {~=~}
                \int_use:N \l_tmpa_int
                ,~
              }
            \tl_set:Nn \l_tmpa_tl { ##1 }
            \int_zero:N \l_tmpa_int
          }
        \int_compare:nT { \l_tmpb_int == \clist_count:N \l_tmpa_clist }
          {
            ##1 {~=~}
            \int_incr:N \l_tmpa_int
            \int_use:N \l_tmpa_int
          }
      }
  }

\NewDocumentCommand \testcmd { m }
  {
    \testcmd_fn:n { #1 }
  }
\ExplSyntaxOff
```

입력: `\testcmd{abaracadabra}`
출력: a = 6, b = 2, c = 1, d = 1, r = 2