Assignment 1

z5141448 Ruofei HUANG

April 9, 2018

Definations

Two Dimensional Arrays Expression

To describe variants of two-dimensional arrays we write $(b : k \mapsto j \mapsto x)$ instead of $(b : k \mapsto (b[k] : j \mapsto x))$. We use this new notation to state an instance of the array-assignment axiom we saw already

$$\left\{\phi[^{(b:k\mapsto x)}/_b]\right\} b[k] := x \left\{\phi\right\}$$

for two-dimensional arrays:

$$\left\{\phi[^{(b:k\mapsto j\mapsto x)}/_b]\right\} b[k][j] := x \left\{\phi\right\}$$

String Length

A string $S \in \mathit{Letter}^*$ which is an array of letters[1].Also, string will be terminate by the null character which is a convention by the C programming language and we will follow this convention in this proof. We write $|S|$ for the number of letters in the string. Formally, we define these two nothion inductively by

$$|S\ell| = |S| + \begin{cases} 1 & \text{if } \ell \neq' \backslash 0' \\ 0 & \text{if } \ell =' \backslash 0' \end{cases}$$

Also, by the convention of C we has this definition for $S \in \mathit{string}$.

$$S[|S|] =' \backslash 0' \wedge \forall 0 \leq i < |S| \, (S[i] \neq' \backslash 0')$$

[1]The letter here is a legal charater encode with ASCII, UTF-8 or other charater encoding standard.

Precondiction

In the toy language's programme, the string length must has a way to calculate. Here is the programme to calculate the String Length

$$a \in String$$

Postcondiction

$$a \in String \wedge aLength = |a|$$

Programme

$$
\begin{array}{l}
\{a \in String\} \\
\{I[^0/_{aLength}]\} \\
aLength := 0; \\
\textbf{while } a[aLength]! =' \backslash 0' \textbf{ do} \\
\quad \{I \wedge a[aLength] \neq' \backslash 0'\} \\
\quad \{I[^{aLength+1}/_{aLength}]\} \\
\quad aLength := saLength + 1; \\
\textbf{od} \\
\{I \wedge a[aLength] =' \backslash 0'\} \\
\{a \in String \wedge aLength = |a|\}
\end{array}
$$

Where

$$I = a \in String \wedge \forall k \in 0..(aLength - 1)\,(a[k] \neq' \backslash 0')$$

First Implication: $a \in String \Rightarrow I[^0/_{aLength}]$

$$
\begin{array}{ll}
 & a \in String \\
\Rightarrow & \quad \langle \text{The } \forall k \in 0..(0-1)\,(a[k] \neq' \backslash 0') \text{ is always true.} \rangle \\
 & a \in String \wedge \forall k \in 0..(0-1)\,(a[k] \neq' \backslash 0') \\
\Leftrightarrow & \quad \langle \text{Defination of I.} \rangle \\
 & I[^0/_{aLength}]
\end{array}
$$

Second Implication: $I \wedge a[aLength] \neq' \backslash 0' \Rightarrow I[^{aLength+1}/_{aLength}]$

$$
\begin{aligned}
& I \wedge a[aLength] \neq' \backslash 0' \\
\Leftrightarrow & \qquad \langle \text{Defination of I.} \rangle \\
& a \in String \wedge \forall k \in 0..(aLength-1)\,(a[k] \neq' \backslash 0') \wedge a[aLength] =' \backslash 0' \\
\Leftrightarrow & \qquad \langle \text{Merge same condition of } a[i] \neq' \backslash 0' \rangle \\
& a \in String \wedge \forall k \in 0..(aLength)\,(a[k] \neq' \backslash 0') \\
\Leftrightarrow & \qquad \langle \text{Substitue back of I.} \rangle \\
& I[^{aLength+1}/_{aLength}]
\end{aligned}
$$

Third Implication: $I \wedge a[aLength] =' \backslash 0' \Rightarrow a \in String \wedge aLength = |a|$

$$
\begin{aligned}
& I \wedge a[aLength] =' \backslash 0' \\
\Leftrightarrow & \qquad \langle \text{Definition of I.} \rangle \\
& a \in String \wedge \forall k \in 0..(aLength-1)\,(a[k] \neq' \backslash 0') \wedge a[aLength] =' \backslash 0' \\
\Leftrightarrow & \qquad \langle \text{Definition of postcondiction and the definition of String.} \rangle \\
& a \in String \wedge aLength = |a|
\end{aligned}
$$

String Equals

To describe two string a, b ($a, b \in String$) are equals we write $a = b$ when:

$$a = b \iff |a| = |b| \wedge \forall j \in 0..\,|a|\,(a[j] = b[j])$$

Similarly, we write:

$$a \neq b \iff \neg(a = b)$$

Comparing String

After defining what is string equal, we need a pieces programme in toy language to do the dirty work which could compare two strings. Also we need a flag variable to store the truth value of $a = b$

Precondiction

$$a, b \in String$$

Postcondiction

$$flag = (a = b)$$

Programme

$\{a, b \in String\}$
$\{I[^{\text{TRUE}}/_{flag}][^{0}/_{i}]\}$
$i := 0;$
$flag := 1;$
$\{I\}$
if $|a| = |b|$ **then**
  $\{I \wedge |a| = |b|\}$
  $\{J\}$
  while $i \leq |a|$ **do**
    $\{J \wedge i \leq |a|\}$
    $\{K\}$
    if $a[i] = b[i]$ **then**
      $\{K \wedge a[i] = b[i]\}$
      $skip$
    else
      $\{K \wedge a[i] \neq b[i]\}$
      $\{J[^{\text{FALSE}}/_{flag}][^{i+1}/_{i}]\}$
      $flag := 0;$
      $\{J[^{i+1}/_{i}]\}$
    fi
    $\{J[^{i+1}/_{i}]\}$
    $i := i + 1;$
    $\{J\}$
  od
  $\{J \wedge i > |a|\}$
  $\{I\}$
else
  $\{I \wedge |a| \neq |b|\}$
  $\{I[^{\text{FALSE}}/_{flag}]\}$
  $flag := 0;$

$\{I\}$

fi

$\{I\}$

$\{flag = (a = b)\}$

Where TRUE$= 1$, FALSE$= 0$, and

$$I = a, b \in String \wedge flag = (|a| = |b| \wedge 0 \leq i \leq (|a| + 1) \wedge \forall k \in 0..(i-1)\,(a[i] = b[i]))$$
$$J = a, b \in String \wedge flag = (0 \leq i \leq (|a| + 1) \wedge \forall k \in 0..(i-1)\,(a[i] = b[i]))$$
$$K = \left(\begin{array}{l} a[i] \neq b[i] \Rightarrow I[^{\text{FALSE}}/_{flag}][^{i+1}/_i] \\ a[i] = b[i] \Rightarrow I[^{i+1}/_i] \end{array} \right)$$

First Implication:$a, b \in String \Rightarrow I[^{\textbf{true}}/_{flag}][^0/_i]$

$a, b \in String$

$\Rightarrow$ $\langle$The append junction is alway true.$\rangle$

$a, b \in String \wedge |a| = |b| \wedge 0 \leq i \leq 0 \wedge \forall k \in 0..(i-1)\,(a[i] = b[i])$

$\Rightarrow$ $\langle flag =$ TRUE and TRUE $=$ TRUE$\rangle$

$a, b \in String \wedge flag = (|a| = |b| \wedge 0 \leq i \leq 0 \wedge \forall k \in 0..(i-1)\,(a[i] = b[i]))$

$\Leftrightarrow$ $\langle$Defination of I.$\rangle$

$I[^{\text{TRUE}}/_{flag}]$

Second Implication:$I \wedge |a| = |b| \Rightarrow J \wedge [^0/_i]$

$I \wedge |a| = |b|$

$\Rightarrow$ $\langle$Definiation of I. $i = 0$ when right junction is always true.$\rangle$

$a, b \in String \wedge flag = (|a| = |b| \wedge 0 \leq 0 \leq (|a| + 1)$
$\wedge \forall k \in 0..(i-1)\,(a[i] = b[i]))$

$\Rightarrow$ $\langle$Definition of J.$\rangle$

$J \wedge [^0/_i]$

Third Implication: $J \wedge i \leq |a| \Rightarrow K$

$$
\begin{array}{ll}
& J \wedge i \leq |a| \\
\Leftrightarrow & \quad \langle \text{Definition of J} \rangle \\
& a, b \in String \wedge flag = (0 \leq i \leq (|a|+1) \wedge \forall k \in 0..(i-1)\,(a[i] = b[i])) \wedge i \leq |a| \\
\Rightarrow & \quad \langle \text{By definition of two string is equal, two case to consider. } \rangle \\
& \left(\begin{array}{l} a[i] \neq b[i] \Rightarrow a, b \in String \wedge flag = \text{FALSE} \\ a[i] = b[i] \Rightarrow a, b \in String \wedge flag = (|a| = |b| \\ \wedge 0 \leq i+1 \leq (|a|+2) \wedge \forall k \in 0..i\,(a[i+1] = b[i+1])) \wedge i \leq |a| \end{array} \right) \\
\Rightarrow & \quad \langle \text{By definition of two string is equal, two case to consider. } \rangle \\
& \left(\begin{array}{l} a[i] \neq b[i] \Rightarrow a, b \in String \wedge flag = \text{FALSE} \\ a[i] = b[i] \Rightarrow a, b \in String \wedge flag = (|a| = |b| \\ \wedge 0 \leq i+1 \leq (|a|+1) \wedge \forall k \in 0..i\,(a[i+1] = b[i+1])) \end{array} \right) \\
\Leftrightarrow & \quad \langle \text{Definition of } I \rangle \\
& \left(\begin{array}{l} a[i] \neq b[i] \Rightarrow I[^{\text{FALSE}}/_{flag}][^{i+1}/_{i}] \\ a[i] = b[i] \Rightarrow I[^{i+1}/_{i}] \end{array} \right) \\
\Leftrightarrow & \quad \langle \text{Defination of } K \rangle \\
& K
\end{array}
$$

Forth Implication: $I \wedge |a| \neq |b| \Rightarrow I[^{\textsf{false}}/_{flag}]$

$$
\begin{array}{ll}
& I \wedge |a| \neq |b| \\
\Rightarrow & \quad \langle \text{Definition of two string is not equal.} \rangle \\
& I \wedge |a| \neq |b| \wedge a \neq b \\
\Rightarrow & \quad \langle flag = \text{FALSE and FALSE} = \text{FALSE} \rangle \\
& a, b \in String \wedge flag = (a = b) \wedge a \neq b \\
\Leftrightarrow & \quad \langle \text{Definition of I.} \rangle \\
& I[^{\text{FALSE}}/_{flag}]
\end{array}
$$

Fifth Implication: $I \Rightarrow flag = (a = b)$

$$
\begin{array}{ll}
& I \\
\Leftrightarrow & \quad \langle \text{Defination of the purpose of } flag \rangle \\
& flag = (a = b)
\end{array}
$$

String Assign

To assign a string to another string array, we will denote as

$$a := b$$

instead of a long programme of our toy language:

$\{a, b \in String\}$
$\{I[^0/_i]\}$
$i := 0;$
$\{I\}$
while $i \leq |b|$ **do**
  $\{I \wedge i \leq |b|\}$
  $\{I[^{i+1}/_i][^{a:i \mapsto b[i]}/_a]\}$
  $a[i] := b[i];$
  $\{I[^{i+1}/_i]\}$
  $i := i + 1;$
  $\{I\}$
od;
$\{I \wedge i > |b|\}$
$\{a, b \in String \wedge a = b\}$

when our invariant is

$$I = a, b \in String \wedge 0 \leq i \leq (|b| + 1) \wedge \forall k \in 0..(i-1)\,(a[k] = b[k])$$

Here are the proofs of the implications:

First Implication for String assign: $a, b \in String \Rightarrow I[^0/_i]$

$$\begin{array}{ll}
 & a, b \in String \\
\Rightarrow & \langle \text{using } |b| \in \mathbb{N} \text{ and realising that the last conjunct is vacuously true} \rangle \\
 & a, b \in String \wedge 0 \leq 0 \leq (|b| + 1) \wedge \forall k \in 0..(0-1)\,(a[k] = b[k]) \\
\Leftrightarrow & \langle \text{definition of I and substitution} \rangle \\
 & I[^0/_i]
\end{array}$$

Second Implication $I \wedge i \leq |b| \Rightarrow I[^{i+1}/_i][^{a:i \mapsto b[i]}/_a]$

We first look at the LHS:

$$\begin{array}{ll}
 & I \wedge i \leq |b| \\
\Leftrightarrow & \langle \text{Substitue I } \rangle \\
 & a, b \in String \wedge 0 \leq i \leq (|b| + 1) \wedge \forall k \in 0..(i-1)\,(a[k] = b[k]) \wedge i \leq |b| \\
\Leftrightarrow & \langle \text{Conjunct } i \leq (|b| + 1) \text{ and } i \leq |b| \rangle \\
 & a, b \in String \wedge 0 \leq i \leq |b| \wedge \forall k \in 0..(i-1)\,(a[k] = b[k])
\end{array}$$

We then expand RHS:

$$
\begin{aligned}
& I[{}^{i+1}/_{i}][{}^{a:i\mapsto b[i]}/_{a}] \\
\Leftrightarrow \quad & \langle \text{substitute } i = i+1 \text{ and } a[i] = b[i] \text{ by definition} \rangle \\
& a, b \in String \wedge 0 \leq i+1 \leq (|b|+1) \wedge \forall k \in 0..((i+1)-1)\,(a[k] = b[k]) \wedge a[i] := b[i]
\end{aligned}
$$

We then have a clear imply

$$
\begin{aligned}
& a, b \in String \wedge 0 \leq i \leq |b| \wedge \forall k \in 0..(i-1)\,(a[k] = b[k]) \\
\Rightarrow \quad & \langle i \leq |b| \Rightarrow i+1 \leq |b|+1 \text{ and } a[i] := b[i] \ \rangle \\
& a, b \in String \wedge 0 \leq i+1 \leq (|b|+1) \wedge \forall k \in 0..((i+1)-1)\,(a[k] = b[k]) \wedge a[i] := b[i]
\end{aligned}
$$

Third Implication $I \wedge i > |b| \Rightarrow a, b \in String \wedge a = b$

$$
\begin{aligned}
& I \wedge i > |b| \\
\Leftrightarrow \quad & \langle \text{substitution of I } \rangle \\
& a, b \in String \wedge 0 \leq i \leq (|b|+1) \wedge \forall k \in 0..(i-1)\,(a[k] := b[k]) \wedge i > |b| \\
\Leftrightarrow \quad & \langle\ i > |b| \text{and } i \leq (|b|+1) \text{ with some calculation} \rangle \\
& a, b \in String \wedge \forall k \in 0..\,|b|\,(a[k] := b[k]) \\
\Rightarrow \quad & \langle \text{Definition of two string equal} \rangle \\
& a, b \in String \wedge a = b
\end{aligned}
$$

1 Task 1

Since we have define some manipulation of String, we can see a string as a whole. So the input is an array of String. Also , the ouput is store in b which is an empty array (type is $String*$ too). Hence we can define our precondiction as:

$$a, b \in String^* \wedge |a| = n$$

As the post condition as:

$$a, b \in String \wedge \forall i < n\,(a[i] = b[m(a, i)])$$

Where m is a mapping function define recursively as follow:

$$
m(a, i) = \begin{cases} -1 & \text{if } i < 0 \\ 0 & \text{if } i = 0 \\ m(a, i-1) & \text{if } a[i] = a[i-1] \\ m(a, i-1) + 1 & \text{if } a[i] \neq a[i-1] \end{cases}
$$

2 Task 2

We propose the following proof outline to demonstrate the correctness of our code (in black).

$\{a, b \in String^* \wedge |a| = n\}$ (1)
$\{J\}$ (2)
if $|a| > 0$ **then** (3)
  $\{J \wedge |a| > 0\}$ (4)
  $\{I[{}^{1}/_{j}][{}^{1}/_{i}][{}^{b:0\mapsto a[0]}/_{b}]\}$ (5)
  $b[0] := a[0];$ (6)
  $\{I[{}^{1}/_{j}][{}^{1}/_{i}]\}$ (7)
  $i = 1; j = 1;$ (8)
  $\{I\}$ (9)
else (10)
  $\{J \wedge |a| \leq 0\}$ (11)
  $\{I[{}^{0}/_{i}][{}^{0}/_{j}]\}$ (12)
  $i := 0; j := 0;$ (13)
  $\{I\}$ (14)
fi (15)
$\{I\}$ (16)
while $i < |a|$ **do** (17)
  $\{I \wedge i < |a|\}$ (18)
  $\{K\}$ (19)
  if $a[i] \neq a[i-1]$ **then** (20)
    $\{K \wedge a[i] \neq a[i-1]\}$ (21)
    $\{I[{}^{i+1}/_{i}][{}^{j+1}/_{j}][{}^{b:j\mapsto a[i]}/_{b}]\}$ (22)
    $b[j] := a[i];$ (23)
    $\{I[{}^{i+1}/_{i}][{}^{j+1}/_{j}]\}$ (24)
    $j := j + 1;$ (25)
    $\{I[{}^{i+1}/_{i}]\}$ (26)
  else (27)
    skip (28)
  fi (29)
  $\{I[{}^{i+1}/_{i}]\}$ (30)
  $i := i + 1$ (31)

$$\{I\} \tag{32}$$

$$\textbf{od} \tag{33}$$

$$\{I \wedge i \geq |a|\} \tag{34}$$

$$\{a, b \in String \wedge \forall i < n\, (a[i] = b[m(a, i)])\} \tag{35}$$

Here are the invariants of this programme [2]:

$$I = a, b \in String^* \wedge |a| = n \wedge 0 \leq i \leq |a| \wedge \forall k \in 0..(i-1)\, (a[k] = b[m(a, k)])$$
$$\wedge j = |b| = m(a, i-1) + 1$$

$$J = \left(\begin{array}{l} |a| > 0 \Rightarrow I[^1/_j][^1/_i][^{b:0 \mapsto a[0]}/_b] \\ |a| \leq 0 \Rightarrow I[^0/_j][^0/_i] \end{array} \right)$$

$$K = \left(\begin{array}{l} a[i] \neq a[i-1] \Rightarrow I[^{i+1}/_i][^{j+1}/_j][^{b:j \mapsto a[i]}/_b] \\ a[i] = a[i-1] \Rightarrow I[^{i+1}/_i] \end{array} \right)$$

2.1 First Implication: $a, b \in String^* \wedge |a| = n \Rightarrow J$

$$a, b \in String^* \wedge |a| = n$$

$\Rightarrow$ $\langle$The right append junction is always true. $\rangle$

$$\left(\begin{array}{l} |a| > 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 1 \leq |a| \wedge \\ \forall k \in 0..(1-1)\, (a[k] = b[m(a, k)]) \wedge j = |b| = m(a, i-1) + 1 \\ |a| = 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 0 \leq |a| \wedge \\ \forall k \in 0..(0-1)\, (a[k] = b[m(a, k)]) \wedge j = |b| = m(a, i-1) + 1 \end{array} \right)$$

$\Rightarrow$ $\langle$Substitution of I $\rangle$

$$\left(\begin{array}{l} |a| > 0 \Rightarrow I[^1/_j][^1/_i][^{b:0 \mapsto a[0]}/_b] \\ |a| \leq 0 \Rightarrow I[^0/_j][^0/_i] \end{array} \right)$$

$\Leftrightarrow$ $\langle$Definition of J$\rangle$

$$J$$

[2]The invariant is following the case study in week 8, might not be true for the stuff we study for now. But I have to use this tool otherwise I couldn't countinue this proof

2.2 Second Implication: $I \wedge i < |a| \Rightarrow K$

$$
\begin{aligned}
& I \wedge i < |a| \\
\Leftrightarrow \quad & \langle \text{Definition of } I \rangle \\
& a, b \in String^* \wedge |a| = n \wedge 0 \le i \le |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \wedge i < |a| \\
\Rightarrow \quad & \langle m(a,i) \text{ need to consider two situation of } a[i] \; (a[i] = a[i-1] \text{ and } a[i] \ne a[i-1]) \rangle \\
& \begin{pmatrix}
a[i] \ne a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[j] \\
a[i] = a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[j-1]
\end{pmatrix} \\
\Leftrightarrow \quad & \langle \text{By the recursively definition of } m(a,i) \rangle \\
& \begin{pmatrix}
a[i] \ne a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[m(a,i)] \\
a[i] = a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[m(a,i)]
\end{pmatrix} \\
\Leftrightarrow \quad & \langle \text{Merge two cases of } k \; (k = i \text{ and } k \in 0..(i-1)) \text{ .} \rangle \\
& \begin{pmatrix}
a[i] \ne a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i)\,(a[k] = b[m(a,k)]) \\
a[i] = a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \le i+1 \le |a| \wedge \forall k \in 0..(i)\,(a[k] = b[m(a,k)])
\end{pmatrix} \\
\Leftrightarrow \quad & \langle \text{Substitue back by definition of I.} \rangle \\
& \begin{pmatrix}
a[i] \ne a[i-1] \Rightarrow I[^{i+1}/_{i}][^{j+1}/_{j}][^{b:j \mapsto a[i]}/_{b}] \\
a[i] = a[i-1] \Rightarrow I[^{i+1}/_{i}]
\end{pmatrix} \\
\Leftrightarrow \quad & \langle \text{Definition of K} \rangle \\
& K
\end{aligned}
$$

The meaning of "By the recursively definition of $m(a,i)$" is when $a[i] = a[i-1]$, $a[i] = b[m(a,i)] = b[m(a,i-1)]$, which $m(a,i-1) = j-1$ since $j = |b|$. In another case, $a[i] \ne a[i-1]$, $b[j] = a[i]$ (where $|b| = j+1$)

2.3 Third Implication:

$$I \wedge i \geq |a| \Rightarrow a, b \in String \wedge \forall i < n\,(a[i] = b[m(a,i)])$$

$$
\begin{aligned}
& I \wedge i \geq |a| \\
\Leftrightarrow \quad & \langle \text{Definations of I} \rangle \\
& a, b \in String^* \wedge |a| = n \wedge 0 \leq i \leq |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \wedge i \geq |a| \\
\Leftrightarrow \quad & \langle i \leq |a| \wedge i \geq |a| \Leftrightarrow i = |a| \rangle \\
& a, b \in String^* \wedge |a| = n \wedge \forall k \in 0..(|a|-1)\,(a[k] = b[m(a,k)]) \\
\Rightarrow \quad & \langle |a| = n \text{ and } i \in \mathbb{N} \rangle \\
& a, b \in String^* \wedge \forall i \in 0..(n-1)\,(a[i] = b[m(a,i)]) \\
\Leftrightarrow \quad & \langle \text{Defination of post condition.} \rangle \\
& a, b \in String \wedge \forall i < n\,(a[i] = b[m(a,i)])
\end{aligned}
$$

3 Task 3

```
int length_str(char *a){
    int i = 0;
    while(a[i]!='\0'){
        i = i+1;
    }
    return i;
}

void copy_str(char *a, char *b){
    // copy a particular string from a to b
    int i= 0;
    while(i <= length_str(a) ){
        b[i] = a[i];
        i++;
    }
}

int compare_str(char *a, char *b){
    int flag = 1;
    if(length_str(a) == length_str(b))
    {
        int i =0;
        while (i <= length_str(a)){
            if (a[i]!= b[i]){
```

```
                flag = 0;
            }
            else{
                // skip
            }
            i++;
        }
    }
    else{
        flag =0;
    }
    return flag;
}

unsigned int uniq(unsigned int n, char *a[], char *b[]){
    int i,j;
    if(n > 0){
        copy_str(a[0],b[0]);
        i =j =1;
    }
    else{
        i = j=0;
    }
    while(i<n){
        if (!compare_str(a[i], a[i-1])){
            copy_str(a[i],b[j]);
            j++;
        }
        else{
            // skip
        }
        i++;
    }
    return j;
}
```

4 Task 4