Assignment 1

z5141448 Ruofei HUANG

April 10, 2018

Defination

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

$\{a \in String\}$
$\{I[^0/_{aLength}]\}$
$aLength := 0;$
while $a[aLength] \neq' \backslash 0'$ **do**
  $\{I \wedge a[aLength] \neq' \backslash 0'\}$
  $\{I[^{aLength+1}/_{aLength}]\}$
  $aLength := saLength + 1;$
od
$\{I \wedge a[aLength] =' \backslash 0'\}$
$\{a \in String \wedge aLength = |a|\}$

Where

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

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

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

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

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

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

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

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

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

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.$\rangle$

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

$\Rightarrow$ $\langle flag = \text{TRUE}$ 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: $J \wedge i \leq |a| \Rightarrow K$

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

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

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

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

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

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

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

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

We first look at the LHS:

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

We then expand RHS:

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

We then have a clear imply

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

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

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

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:

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

As the post condition as:

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

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)

$\{I\}$ (32)

od (33)

$\{I \wedge i \geq |a|\}$ (34)

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

Here are the invariants of this programme [2]:

$$I = a, b \in String^* \wedge |a| = n \wedge 0 \leq i \leq |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)])$$
$$\wedge j = m(a, i-1) + 1$$
$$J = \begin{pmatrix} |a| > 0 \Rightarrow I[{}^{1}/_{j}][{}^{1}/_{i}][{}^{b:0\mapsto a[0]}/_{b}] \\ |a| \leq 0 \Rightarrow I[{}^{0}/_{j}][{}^{0}/_{i}] \end{pmatrix}$$
$$K = \begin{pmatrix} a[i] \neq a[i-1] \Rightarrow I[{}^{i+1}/_{i}][{}^{j+1}/_{j}][{}^{b:j\mapsto a[i]}/_{b}] \\ a[i] = a[i-1] \Rightarrow I[{}^{i+1}/_{i}] \end{pmatrix}$$

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

$$\begin{array}{ll} & a, b \in String^* \wedge |a| = n \\ \Rightarrow & \langle \text{The right append junction is always true. } \rangle \\ & \left(\begin{array}{l} |a| > 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 1 \leq |a| \wedge \\ \forall k \in 0..(1-1)\,(a[k] = b[m(a,k)]) \wedge j = 1 \wedge m(a,0) + 1 = 1 \\ |a| = 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 0 \leq |a| \wedge \\ \forall k \in 0..(0-1)\,(a[k] = b[m(a,k)]) \wedge j = 0 \wedge m(a,-1) + 1 = 0 \end{array} \right) \\ \Rightarrow & \langle \text{The number is same.} \rangle \\ & \left(\begin{array}{l} |a| > 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 1 \leq |a| \wedge \\ \forall k \in 0..(1-1)\,(a[k] = b[m(a,k)]) \wedge j = m(a,0) + 1 \\ |a| = 0 \Rightarrow a, b \in String^* \wedge |a| = n \wedge 0 \leq 0 \leq |a| \wedge \\ \forall k \in 0..(0-1)\,(a[k] = b[m(a,k)]) \wedge j = m(a,-1) + 1 \end{array} \right) \\ \Rightarrow & \langle \text{Substitution of I } \rangle \\ & \left(\begin{array}{l} |a| > 0 \Rightarrow I[{}^{1}/_{j}][{}^{1}/_{i}][{}^{b:0\mapsto a[0]}/_{b}] \\ |a| \leq 0 \Rightarrow I[{}^{0}/_{j}][{}^{0}/_{i}] \end{array} \right) \\ \Leftrightarrow & \langle \text{Definition of J} \rangle \\ & J \end{array}$$

[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{array}{ll}
& I \wedge i < |a| \\
\Leftrightarrow & \quad \langle \text{Definition 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\, j = m(a, i-1) + 1 \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] \neq a[i-1]) \rangle \\
& \left(\begin{array}{l}
a[i] \neq a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \leq i+1 \leq |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[j] \wedge j = m(a, (i+1)-1) + 1 \\
a[i] = a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \leq i+1 \leq |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[j-1] \wedge j = m(a, (i+1)-1) + 1
\end{array} \right) \\
\Leftrightarrow & \quad \langle \text{By the recursively definition of } m(a,i) \rangle \\
& \left(\begin{array}{l}
a[i] \neq a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \leq i+1 \leq |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 \leq i+1 \leq |a| \wedge \forall k \in 0..(i-1)\,(a[k] = b[m(a,k)]) \\
\wedge a[i] = b[m(a,i)]
\end{array} \right) \\
\Leftrightarrow & \quad \langle \text{Merge two cases of } k \ (k = i \text{ and } k \in 0..(i-1)) \ . \rangle \\
& \left(\begin{array}{l}
a[i] \neq a[i-1] \Rightarrow a, b \in String^* \wedge |a| = n \wedge \\
0 \leq i+1 \leq |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 \leq i+1 \leq |a| \wedge \forall k \in 0..(i)\,(a[k] = b[m(a,k)])
\end{array} \right) \\
\Leftrightarrow & \quad \langle \text{Substitue back by definition of I.} \rangle \\
& \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) \\
\Leftrightarrow & \quad \langle \text{Definition of K} \rangle \\
& K
\end{array}
$$

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, $a[i] \neq a[i-1]$, $b[j] = a[i]$ (where $|b| = j+1$)

2.3 Third Implication:

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

$I \wedge i \geq |a|$

$\Leftrightarrow$ $\langle$Definations of I$\rangle$

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

$\Leftrightarrow$ $\langle i \leq |a| \wedge i \geq |a| \Leftrightarrow i = |a| \rangle$

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

$\Rightarrow$ $\langle |a| = n$ and $i \in \mathbb{N} \rangle$

$a, b \in String^* \wedge \forall i \in 0..(n-1)\,(a[i] = b[m(a,i)])$

$\Leftrightarrow$ $\langle$Defination of post condition.$\rangle$

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

3 Task 3

3.1 Unsatisfied Programme

```
1  int length_str(char *a){
2      int i = 0;
3      while(a[i]!='\0'){
4          i = i+1;
5      }
6      return i;
7  }
8
9  void copy_str(char *a, char *b){
10     // copy a particular string from a to b
11     int i= 0;
12     while(i <= length_str(a) ){
13         b[i] = a[i];
14         i++;
15     }
16 }
17
18 int compare_str(char *a, char *b){
19     int flag = 1;
20     if(length_str(a) == length_str(b))
21     {
```

```
22          int i =0;
23          while (i <= length_str(a)){
24              if (a[i]!= b[i]){
25                  flag = 0;
26              }
27              else{
28                  // skip
29              }
30              i++;
31          }
32      }
33      else{
34          flag =0;
35      }
36      return flag;
37  }
38
39  unsigned int uniq(unsigned int n, char *a[], char *b[]){
40      int i,j;
41      if(n > 0){
42          copy_str(a[0],b[0]);
43          i =j =1;
44      }
45      else{
46          i = j=0;
47      }
48      while(i<n){
49          if (!compare_str(a[i], a[i-1])){
50              copy_str(a[i],b[j]);
51              j++;
52          }
53          else{
54              // skip
55          }
56          i++;
57      }
58      return j;
59  }
```

By the building of this programme, we build a programme that meet the criteria of our toy language. For the validity of this programme, we already have similar proof given by this article, since we have proof the parts seperatly.

3.2 Satisfied Programme in Toy Language Style

3.2.1 Precondiction

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

3.2.2 Postcondiction

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

Where m is a mapping function define recursively as follow:

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

3.3 Toy Language Programme

$\{a, b \in String^* \wedge |a| = n\}$ (36)
$\{I\}$ (37)
if $|a| > 0$ **then** (38)
  $\{I \wedge |a| > 0\}$ (39)
  $\{J[^0/_{aLength}]\}$ (40)
  $aLength := 0;$ (41)
  $\{J\}$ (42)
  while $a[0][aLength] \neq' \backslash 0'$ **do** (43)
    $\{J \wedge a[0][aLength] \neq' \backslash 0'\}$ (44)
    $\{J[^{k+1}/_k]\}$ (45)
    $k := k + 1$ (46)
    $\{J\}$ (47)
  od (48)

$\{J \wedge a[0][aLength] =' \backslash 0'\}$ (49)

$\{K[^0/_k]\}$ (50)

$k := 0;$ (51)

$\{K\}$ (52)

while $k \leq |a[0]|$ **do** (53)

$\{K \wedge k \leq |a[0]|\}$ (54)

$\{K[^{k+1}/_k][^{(b:0 \mapsto k \mapsto a[0][k])}/_b]\}$ (55)

$b[0][k] := a[0][k]$ (56)

$\{K[^{k+1}/_k]\}$ (57)

$k := k + 1;$ (58)

$\{K\}$ (59)

od (60)

$\{K \wedge k > |a[0]|\}$ (61)

$\{L[^1/_i][^1/_j]\}$ (62)

$i := 1; j := 1;$ (63)

$\{L\}$ (64)

else (65)

$\{I \wedge |a| \leq 0\}$ (66)

$\{L[^0/_i][^0/_j]\}$ (67)

$i := 0; j := 0;$ (68)

$\{L\}$ (69)

fi (70)

$\{L\}$ (71)

while $i < |a|$ **do** (72)

$\{L \wedge i < |a|\}$ (73)

$\{M[^0/_{aLength}]\}$ (74)

$aLength := 0;$ (75)

$\{M\}$ (76)

while $a[i][aLength] \neq' \backslash 0'$ **do** (77)

$\{M \wedge a[i][aLength] \neq' \backslash 0'\}$ (78)

$\{M[^{aLength+1}/_{aLength}]\}$ (79)

$aLength = aLength + 1;$ (80)

$\{M\}$ (81)

od (82)

$\{N \wedge a[i-1][bLength] =' \backslash 0'\}$ (83)

$\{N[^{0}/_{bLength}]\}$ (84)

$bLength := 0;$ (85)

$\{N\}$ (86)

while $a[i-1][bLength] \neq' \backslash 0'$ **do** (87)

  $\{N \wedge a[i-1][bLength] \neq' \backslash 0'\}$ (88)

  $\{N[^{bLength+1}/_{bLength}]\}$ (89)

  $bLength = bLength + 1;$ (90)

  $\{N\}$ (91)

od (92)

$\{N \wedge a[i-1][bLength] =' \backslash 0'\}$ (93)

$\{O\}$ (94)

if $aLength = bLength$ **then** (95)

  $\{O \wedge aLength = bLength\}$ (96)

  $\{P[^{0}/_{k}]\}$ (97)

  $k := 0;$ (98)

  $\{P\}$ (99)

  while $k \leq aLength$ **do** (100)

    $\{P \wedge k \leq aLength\}$ (101)

    $\{Q\}$ (102)

    if $a[i][k] \neq a[i-1][k]$ **then** (103)

      $\{Q \wedge a[i][k] \neq a[i-1][k]\}$ (104)

      $\{P[^{k+1}/_{k}][^{\text{FALSE}}/_{flag}]\}$ (105)

      $flag := 0;$ (106)

      $\{P[^{k+1}/_{k}]\}$ (107)

    else (108)

      $\{P[^{k+1}/_{k}]\}$ (109)

      $skip$ (110)

    fi (111)

    $\{P[^{k+1}/_{k}]\}$ (112)

    $k := k + 1;$ (113)

    $\{P\}$ (114)

  od (115)

  $\{P\}$ (116)

  $\{O\}$ (117)

then (118)

$\{O \wedge aLength \neq bLength\}$ (119)

$\{O[^{\text{FALSE}}/_{flag}]\}$ (120)

$flag := 0;$ (121)

$\{O\}$ (122)

fi (123)

$\{O\}$ (124)

$\{R\}$ (125)

if $flag \neq$ TRUE **then** (126)

$\{R \wedge flag \neq \text{TRUE}\}$ (127)

$\{S[^{0}/_{k}]\}$ (128)

$k := 0;$ (129)

$\{S\}$ (130)

while $k \leq aLength$ **do** (131)

$\{S \wedge k \leq aLength\}$ (132)

$\{S[^{k+1}/_{K}][^{(b:j \mapsto j \mapsto a[i][k])}/_{b}]\}$ (133)

$b[j][k] = a[i][k];$ (134)

$k := k + 1;$ (135)

$\{S\}$ (136)

od (137)

$\{S \wedge k > aLength\}$ (138)

$\{R[^{j+1}/_{j}]\}$ (139)

$j := j + 1;$ (140)

$\{R\}$ (141)

else (142)

$\{R \wedge flag = \text{TRUE}\}$ (143)

$skip$ (144)

fi (145)

$\{R\}$ (146)

$\{L[^{i+1}/_{i}]\}$ (147)

$i := i + 1;$ (148)

$\{L\}$ (149)

od (150)

$\{L \wedge i \geq |a|\}$ (151)

$\{a, b \in String \wedge \forall i < n\,(a[i] = b[m(a, i)])\}$ (152)

I won't give the complete proof of this programme, since all the seperatly part and the

correct model has been proofed in the previous section. If you can follow the proof by task 2, then this programme would be true.

3.4 Programme in C

```
1  unsigned int uniq(unsigned int n, char *a[], char *b[]){
2      int i,j,k;
3      // store the length of String
4      int aLength, bLength,flag;
5      if(n > 0){
6          aLength= 0;
7
8          while(a[0][aLength]!='\0'){
9              aLength = aLength+1;
10         }
11
12         k= 0;
13         while(k <= aLength ){
14                 b[0][k] = a[0][k];
15                 k= k+1;
16         }
17         i =j =1;
18     }
19     else{
20         i = j=0;
21     }
22     while(i<n){
23         aLength =bLength = 0;
24         while(a[i][aLength]!='\0'){
25             aLength = aLength+1;
26         }
27         while(a[i-1][bLength]!='\0'){
28             bLength = bLength+1;
29         }
30         flag = 1;
31         if(aLength == bLength)
32         {
33             k =0;
34             while (k <= aLength){
35                 if (a[i][k]!= a[i-1][k]){
36                     flag = 0;
37                 }
38                 else{
```

```
39                      // skip
40                  }
41                  k= k+1;
42              }
43          }
44          else{
45              flag =0;
46          }
47          if (!flag){
48              k= 0;
49              while(k <= aLength){
50                  b[j][k] = a[i][k];
51                  k= k+1;
52              }
53              j++;
54          }
55          else{
56              // skip
57          }
58          i++;
59      }
60      return j;
61  }
```

4 Task 4

I use the divide and conquer methodology to proof this big programme. I proof the string, String Compare, String Lenth, String Copy. And use all the proofed programme to build the final uniq programme. Since we can not use function call in our toy language, I have supplied a toy language style programme that intergrated all those function in it. I am sure that my proof is not perfect, and skill of using LaTeX is not statisfy. But I try my best on this Assignment. I feel exhausted by this assignment, and put almost all my vacation on this stuff. And feel so unfair about we must do a lot of work to complete the proof, but previous year just a piece of cake compare to this assignment.