COMP2111 - Assignment 1

z5162498 - Bailey Ivancic

April, 2018

1 Task 1

The task for this assignment is to specify and subsequently implement a simplified version of the unix *uniq* command.

The task has the following requirements:

- An array of strings **a** contains a number of elements **n**
- Output is stored in array of strings **b**. Output is the same as **a**, but with adjacent identical strings collapsed into one
- The number of strings in **b** is stored in variable **k**
- Array **a** should not be changed, along with the number of items in **a** (**n**)

Making these requirements into a program specification, we get the precondition:

$$(n \geq 0) \ \wedge (a[n] = 0)$$

stating that array **a** needs to be a 2D array, with the final element of the array being the null character.

From the spec, it can be assumed that all strings provided will be null-terminated, and as such do not form part of the precondition.

and the postcondition:

$$\forall i < n,\ a[i] = b[m(i)],$$
$$where\ m : \mathbb{N} \longrightarrow \mathbb{N} :$$
$$\forall i < n, \exists r \leq k\ suchthat\ (b[r] = a[i] \wedge (\forall k \in \mathbb{N},\ a[i] = a[i + j] \implies k = i)$$

stating that for any index i in a, there exists an index k in b such that b[k] = a[i]. Additionally, if a[i] = a[i + *some natural number*], then k = i.

2 Task 2

We first begin with defining a proof outline for the main program, running under the assumption that strcmp and strcpy are built-in toy-language features. This is done to derive the Invariant and assertions for the main program before bringing in the other functions.

Main:

$\{n \geq 0 \wedge a[n] = 0\}$
$\{I[^{0}/_{temp}]\}$
$temp := 0;$
$\{I[^{0}/_{i}]\}$
$i := 0;$
$\{I[^{0}/_{k}]\}$
$k := 0;$
$\{I\}$
while $i < n$ **do**
  $\{I \wedge i < n\}$
  if $\neg strcmp(a[i], temp)$ *(1)*
    $\{I \wedge a[i] = temp\}$
    $\{I[^{i+1}/_{i}][^{k+1}/_{k}][^{a[i]}/_{b[k]}]\}$
    $strcpy(b[k], a[i]);$ *(2)*
    $\{I[^{i+1}/_{i}][^{a[i]}/_{temp}]\}$
    $strcpy(temp, a[i]);$ *(3)*
    $\{I[^{k+1}/_{k}]\}$
    $k := k + 1;$
  fi
  $\{I[^{i+1}/_{i}]\}$
  $i := i + 1;$
$\{I\}$
od
$\{I \wedge i \geq n\}$
$\{\forall i < n, a[i] = b[m(i)]\}$

From looking at the proof outline, the following invariant $\boldsymbol{I}$ can be derived:

$$I = n = n_0 \wedge a = a_0 \wedge k \leq i$$

Stating that $\boldsymbol{n}$ can never be changed from its original value, the array $\boldsymbol{a}$ is never changed, and $\boldsymbol{k}$ must be $\leq$ to $\boldsymbol{i}$, since it is only incrememnted when $\boldsymbol{i}$ is incremented, and in some iterations never incremented at all.

Following this, we provide proof outlines for the toy-language variants of strcmp and strcpy. There is two versions of strpy, which deal with different arrays, and these are included below as indicated in the main program. This is in order to determine the invariants for each.

strcmp(1):

$\{I \wedge i < n\}$
$\{Q[^{0}/_{j}]\}$
$j := 0;$
$\{Q[^{true}/_{same}]\}$
$same := true;$
$\{I \wedge Q\}$
while $a[i][j] \neq 0 \wedge temp[j] \neq 0$ **do**
  $\{Q \wedge I \wedge a[i][j] \neq 0 \wedge temp[j] \neq 0\}$
  if $a[i][j] \neq temp[j]$
    $\{Q \wedge I \wedge a[i][j] \neq temp[j]\}$
    $\{Q[^{false}/_{same}]\}$
    $same := false;$
  fi
  $\{Q[^{j+1}/_{j}]\}$
  $j := j + 1;$
  $\{I \wedge Q\}$
od
$\{a[i][j] = 0 \vee temp[j] = 0\}$
if $a[i][j] \neq 0 \vee temp[j] \neq 0;$
  $\{I \wedge Q \wedge (a[i][j] \neq 0 \vee temp[j] \neq 0)\}$
  $\{Q[^{false}/_{same}]\}$
  $same := false;$
fi
$\{I \wedge Q \wedge i < n\}$

strcpy(2):

$\{I \wedge i < n\}$
$\left\{F\left[{}^{0}/_{w}\right]\right\}$
$w := 0;$
$\{I \wedge F\}$
while $a[i][w] \neq 0$ **do**
  $\{I \wedge F \wedge a[i][w] \neq 0\}$
  $\left\{F\left[{}^{w+1}/_{w}\right]\left[{}^{a:i \mapsto w \mapsto b[k][w]}/_{a}\right]\right\}$
  $b[k][w] := a[i][w];$
  $\left\{F\left[{}^{w+1}/_{w}\right]\right\}$
  $w := w + 1;$
  $\{I \wedge F\}$
od
$\{I \wedge F \wedge a[i][w] = 0\}$
$\left\{F\left[{}^{b:k \mapsto w \mapsto 0}/_{b}\right]\right\}$
$b[k][w] := 0;$
$\{I \wedge F \wedge i < n\}$
$\{\forall w \in \mathbb{N}, b[k][w] = a[i][w]\}$

strcpy(3):

$\{I \wedge i < n\}$
$\{a[^{0}/_{r}]\}$
$r := 0;$
$\{I \wedge G\}$
while $a[i][r] \neq 0$ **do**
$\{I \wedge G \wedge a[i][r] \neq 0\}$
$\{G[^{r+1}/_{r}][^{temp:r \mapsto a[i][r]}/_{temp}]\}$
$temp[r] := a[i][r];$
$\{G[^{r+1}/_{r}]\}$
$r := r + 1;$
$\{I \wedge G\}$
od
$\{I \wedge G \wedge a[i][r] = 0\}$
$\{G[^{temp:r \mapsto 0}/_{temp}]\}$
$temp[r] := 0;$
$\{I \wedge G \wedge i < n\}$
$\{\forall r \in \mathbb{N}, temp[r] = a[i][r]\}$

From these proof outlines, we can see that the invariants for each function are:

$$Q = (I \wedge a[i] \neq 0 \wedge \forall f \in (0..(j)), a[i][f] \neq temp[f] \implies \neg same)$$
$$F = (I \wedge a[i] \neq 0 \wedge \forall f \in (0..(w)), b[k][f] = a[i][f])$$
$$G = (I \wedge a[i] \neq 0 \wedge \forall f \in (0..(r)), temp[f] = a[i][f])$$

As such, we can simply use the invariant $\boldsymbol{Q}$ to denote the invariants for all three embedded loops. We then propose the proof outline for the whole toy-language program together, with strcmp and strcpy included.

Complete:

$\{n \geq 0 \wedge a[n] = 0\}$
$\{I[^0/_k][^0/_i][^0/_{temp}]\}$
$temp := 0;$
$\{I[^0/_k][^0/_i]\}$
$i := 0;$
$\{I[^0/_k]\}$
$k := 0;$
$\{I\}$
while $i < n$ **do**
  $//Start\ \ of\ \ strcmp(1)$
  $\{I \wedge i < n\}$
  $\{Q[^{true}/_{same}][^0/_j]\}$
  $j := 0;$
  $\{Q[^{true}/_{same}]\}$
  $same := true;$
  $\{I \wedge Q\}$
  while $a[i][j] \neq 0 \wedge temp[j] \neq 0$ **do**
    $\{Q \wedge I \wedge a[i][j] \neq 0 \wedge temp[j] \neq 0\}$
    if $a[i][j] \neq temp[j]$
      $\{Q \wedge I \wedge a[i][j] \neq temp[j]\}$
      $\{Q[^{false}/_{same}]\}$
      $same := false;$
    fi
    $\{Q[^{j+1}/_j]\}$
    $j := j + 1;$
    $\{I \wedge Q\}$
  od
  $\{a[i][j] = 0 \vee temp[j] = 0\}$
  if $a[i][j] \neq 0 \vee temp[j] \neq 0;$
    $\{I \wedge Q \wedge (a[i][j] \neq 0 \vee temp[j] \neq 0)\}$
    $\{Q[^{false}/_{same}]\}$
    $same := false;$
  fi
  $\{I \wedge Q \wedge i < n\}$
  $//End\ \ of\ \ strcmp(1)$

if $\neg same\ (1)$

$\{I \wedge a[i] \neq temp\}$

$//Start\ \ of\ \ strcpy(2)$

$\{F[^{0}/_{w}]\}$

$w := 0;$

$\{I \wedge F\}$

while $a[i][w] \neq 0$ **do**

$\{I \wedge F \wedge a[i][w] \neq 0\}$

$\{F[^{w+1}/_{w}][^{b:k\mapsto w\mapsto a[i][w]}/_{b}]\}$

$b[k][w] := a[i][w];$

$\{F[^{w+1}/_{w}]\}$

$w := w + 1;$

$\{I \wedge F\}$

od

$\{I \wedge F \wedge a[i][w] = 0\}$

$\{F[^{b:k\mapsto w\mapsto 0}/_{b}]\}$

$b[k][w] := 0;$

$\{I \wedge F \wedge i < n\}$

$//End\ \ of\ \ strcpy(2)$

$//Start\ \ of\ \ strcpy(3)$

$\{G[^{0}/_{r}]\}$

$r := 0;$

$\{I \wedge G\}$

while $a[i][r] \neq 0$ **do**

$\{I \wedge G \wedge a[i][r] \neq 0\}$

$\{G[^{r+1}/_{r}][^{temp:r \mapsto a[i][r]}/_{temp}]\}$

$temp[r] := a[i][r];$

$\{G[^{r+1}/_{r}]\}$

$r := r + 1;$

$\{I \wedge G\}$

od

$\{I \wedge G \wedge a[i][r] = 0\}$

$\{G[^{temp:r \mapsto 0}/_{temp}]\}$

$temp[r] := 0;$

$\{I \wedge G \wedge i < n\}$

$//End\ \ of\ \ strcpy(3)$

$\{I[^{k+1}/_{k}]\}$

$k := k + 1;$

fi

$\{I[^{i+1}/_{i}]\}$

$i := i + 1;$

$\{I\}$

od

$\{I \wedge i \geq n\}$

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

Implications

2.1 First implication: $n \geq 0 \wedge a[n] = 0 \implies I[^0/_{temp}]$

$n \geq 0 \wedge a[n] = 0 \implies n = n_0 \wedge a = a_0 \wedge k \leq i$

n and a have not been changed thus far in the program, as such they can be assumed true

$n \geq 0 \wedge a[n] = 0 \implies k \leq 1 \wedge true$

$n \geq 0 \wedge a[n] = 0 \implies 0 \leq 1$

$n \geq 0 \wedge a[n] = 0 \implies true$

Therefore, implication is proved true

2.2 Second implication: $I \wedge i < n \implies Q[^0/_j]$

$n = n_0 \wedge a = a_a \wedge k \leq i \wedge i < n \implies I \wedge a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same$

Expand both LHS and RHS

$i < n \implies a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same$

array a[n] = 0, but we state that i ¡ n. Therefore a[i] $\neq$ 0

$true \implies a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same$

2.3 Third implication: $Q \wedge I \wedge a[i][j] \neq temp[j] \implies Q[^{false}/_{same}]$

$I \wedge a[i] \neq 0 \wedge (\forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same) \wedge (a[i][j] \neq temp[j]) \implies I \wedge a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies true$

Expand both LHS and RHS, while making substitution of false for same on RHS. As such, the rightmost implication becomes true.

$(\forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same) \wedge (a[i][j] \neq temp[j]) \implies \forall f \in (0..j), a[i][f] \neq temp[f] \implies true$

Since I and a[i] $\neq$ 0 appear on both sides, they can be made true on both, cancelling them out.

$(\forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same) \wedge (a[i][j] \neq temp[j]) \implies true$

Since the first portion of the RHS implied true, this means the entire RHS is true

Thus, since the RHS of the conditional statement is true, the entire implication is true

2.4 Fourth implication: $I \wedge Q \wedge (a[i][j] \neq 0 \vee temp[j] \neq 0 \implies Q[^{false}/_{same}]$

$I \wedge I \wedge a[i] \neq 0 \wedge \forall \in (0..j), a[i][f] \neq temp[f] \implies \neg same \implies I \wedge a[i] \neq 0 \wedge \forall f] in (0..j), a[i][f] \neq temp[f] \implies \neg false$

Expanding out the LHS and RHS, and substituting in false for same

$I \wedge a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same \implies I \wedge a[i] \neq 0 \wedge \forall f] in (0..j), a[i][f] \neq temp[f] \implies \neg false$

I ∧ I = I

$I \wedge a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same \implies I \wedge a[i] \neq 0 \wedge \forall f] in (0..j), a[i][f] \neq temp[f] \implies true$

$(I \wedge a[i] \neq 0 \wedge \forall f \in (0..j), a[i][f] \neq temp[f] \implies \neg same) \implies true$

Since the implication in the RHS formulated to true, the RHS is, by extension true.

As a result, the entire implication is true.

2.5 Fifth implication: $I \wedge F \wedge a[i][w] \neq 0 \implies F[^{w+1}/_{w}][^{b:k->w->a[i][w]}/_{b}]$

RHS $= I \wedge (a[i] \neq 0) \wedge (\forall f \in (0..(w+1)), a[k][f] = a[k][f])$

Expanded the RHS, and performed two substitutions

RHS $= I \wedge (a[i] \neq 0) \wedge true$

Last part of the RHS is vacuously true

$I \wedge F \wedge a[i][w] \neq 0 \implies I \wedge a[i][w] \neq 0$

Removing the true to show the remaining elements

$F \wedge true \implies true$

Remaining elements on RHS also appear on LHS, and so RHS becomes true

As a result, the entire implcation is true.

2.6 Fifth implication: $I \wedge F \wedge a[i][w] \neq 0 \implies F[^{b:k->w->0}/_{b}]$

LHS $= I \wedge a[i] \neq 0 \wedge (\forall f \in (0..w), a[k][f] = a[k][f]) \wedge b[k][w] = 0$

RHS $= I \wedge a[i] \neq 0 \wedge (\forall f \in (0..w), a[k][f] = a[k][f])$

Expanded both the LHS and RHS to show similar items

$a[i][w] = 0 \wedge true \implies b[k][w] = 0 \wedge true$

Removed common elements from both sides

Since in F b[k][w] = a[i][w], a[i][w] = 0 does imply that b[k][w] = 0. Therefore, the implication is true

2.7 Sixth implication: $I \wedge G \wedge a[i][r] \neq 0 \implies G[^{temp:r->0}/_{temp}]$

RHS $= I \wedge a[i] \neq 0 \wedge (\forall f \in (0..r), temp[f] = a[i][f]) \wedge temp[r] = 0$
LHS $= I \wedge I \wedge a[i] \neq 0 \wedge (\forall f \in (0..r), temp[f] = a[i][f]) \wedge a[i][r] = 0$
Expand out both LHS and RHS
$true \wedge a[i][r] = 0 \implies true \wedge temp[r] = 0$
Removed common elements from both sides
$a[i][r] = 0 \implies temp[r] = 0$
Since in G temp[f] = a[i][f] and both indexes are set to r, this implication is therefore true

3 Task 3

```
#include "uniq.h"
#include <stdbool.h>
#include <stdio.h>
#define MAXCHAR 1024

unsigned int uniq(unsigned int n, char *a[], char *b[])
{
    char temp[MAXCHAR]; //Last copied string
    int i = 0; //iterate through array a
    unsigned int k = 0; //iterate through array b

    while (i < n)
    {
        //strcmp(a[i], temp) START
        int j = 0;
        bool same = true;
        while (a[i][j] != '\0' && temp[j] != '\0')
        {
            if (a[i][j] != temp[j])
            {
                same = false;
            }
            j = j + 1;
        }

        //One string is longer at end
        if (a[i][j] != '\0' || temp[j] != '\0')
        {
            same = false;
        }
        //strcmp FINISH

        if (!same)
        {
            //strcpy(b[k],a[i] START
            int m = 0;
            while (a[i][m] != '\0')
            {
```

```
                b[k][m] = a[i][m];
                m = m + 1;
            }
            b[k][m] = '\0';
            //strcpy FINISH

            //strcpy(temp, a[i]) START
            m = 0;
            while (a[i][m] != '\0')
            {
                temp[m] = a[i][m];
                m = m + 1;
            }
            temp[m] = '\0';
            //strcpy FINISH

            k = k + 1;
        }
        i = i + 1;
    }
    return k;
}
```

In the C program, the control structures have been kept as close to the toy language as possible to prevent ambiguity. The structure has been kept the same as the toy language program, with the while loops for the strcmp() and strcpy() being put inside the larger while loop to prevent function calls, since these are not a part of the toy langauge.
The string *temp* has been defined with an arbitrary space defined as **MAXCHAR**, to ensure it would be large enough to accomodate any strings that are read into it. This arbitrarily large number will not affect the program functionality/correctness, as all strings are null terminated, thus meaning any extra space will not be used.