CS4350 Group 12 Project 1 Report

Cody Jandt
David Crowe
Karan Mishra
Maria Sanchez
Payton Suiter

Section I - Introduction

We chose to divide this project by having each group member responsible for a menu operation. We used GitHub to share our code changes and GroupMe to communicate amongst each other outside of our group meetings. Payton was responsible for creating the add functionality, Cody was tasked with implementing the update script, Maria was in charge of the remove and display script, Karan was responsible for creating the search feature, and David created the menu interface, implemented edge case testing for the update script and handled multiple found entries in the remove script.

Payton wrote Section I of the report with revision done by David. David wrote Section II of the report and Section III was a collaborative effort between Payton, Maria, Cody, and David. The code portion of Section IV was created by Karan and the screenshots were provided by David.

Section II - Database Design

For our database, we chose to delimit each contact field with a colon and separated each contact in a new line. Each contact has the following fields: Name, Address, Phone, and E-Mail which respectively stores their first and/or last name, where they live, the phone number at which they can be reached, and their electronic mail address.

Below is a sample database with five entries and the aforementioned fields:

```
1 Qijun Gu:209E Comal:2453518:qijun@txstate.edu
2 Rick Sanchez:C-137:1234567:get@schwifty.org
3 Kim Lu:5327 Fun Street:5345845513:kim_lu@mailinator.com
4 Kim Lu:3428 Boring Street:5647382947:kim.lu@etsy.com
5 Monty Python:420 Glasgow Blvd.:8844813841:holy@grail.com
```

Section III - Interface Design & Function

For our database, we created a menu to navigate the database options. This menu was given its own script file. Once the menu function is compiled, the menu display the following options: (a) Add Record, (b) Search Record, (c) Update Record, (d) Remove Record, (e) Display All and (f) Exit. Each option has its own function. The menu script prompts the user to select an option by entering the corresponding key. The menu script then

validates the user input. If an invalid option is selected an error message is displayed, and the user is re-prompted. Once a valid option is selected, the chosen option, the program runs the respective script.

If the user decided to add an entry into the database, the first option would be chosen, which would then run the add script which prompts the user to fill in the contact's name, address, phone number, and email. If any fields contain the delimiter (:) the script would again prompt the user for that field. Once all fields have been successfully read a string containing all the fields is appended into the database file using >>.

If the user wanted to search for an entry, the user would choose the second option to run the search script. The script would ask the user which fields to search for. Once the desired fields are chosen, the user would enter each of the fields. Using grep, the function would then search throughout the database to find the match. If any of the fields exist in the database, the contact information for the field would be displayed. Otherwise, the function would notify the user that the contact was not found.

If the user wanted to update a record, the user would select the third option, which would then run the update function. This script would first ask the user for the name of the contact they wished to update. If there already exists a contact with that name, the user would be prompted to enter a uniquely identifying field, the email. If this email exists in the database, the user is then prompted which fields for the found contact they would like to update. Once all input is received, each of the updated fields would be placed into a variable. The script then removes the outdated contact and appends a new contact with the updated information.

If the user wished to remove an entry, the fourth option would be chosen, which would then run the remove function. This function would ask the user which entry they would like removed. If it was an entry that existed, then it would remove the line that contained the entry. Otherwise, it would echo that the record was not found and returns to the main menu. If multiple records with the same name are found, the user is then asked to enter an email address. Using $grep, the script finds all entries in the database that does not contain the email and

writes to a temporary buffer file which then replaces the contents of the original database. If no matching email is found the user is notified appropriately and the remove script is run again.

If the user wishes to view all records, the fifth option would be chosen, which would then run the display function. This function prompts the user to enter a password before they can view the database to simulate administrator access. If the password is invalid, and an error message will be displayed and return the user to the main function. Once the correct password is entered the entire database will be displayed.

(2) Screenshot to describe one operation of update an existing record step by step

```
Welcome to Contact Database!
**************************
 (a) Add Record
 (b) Search record
 (c) Update record
 (d) Remove Record
 (e) Display Entire Database (Admin Only)
 (f) Exit

Enter choice: e
Enter password:
Showing database..

Osvaldo Phillip  100 Country Rd.  333-444-5555  olly@gmail.com
Osvaldo Phillip  101 Country Rd.  999-888-7777  o_phil@aol.com

Welcome to Contact Database!
**************************
 (a) Add Record
 (b) Search record
 (c) Update record
 (d) Remove Record
 (e) Display Entire Database (Admin Only)
 (f) Exit

Enter choice: c
Please enter the name of the record you would like to update: Osvaldo Phillip
Found more than one matching record. Please enter contact's email: o_phil@aol.com
---
Found contact
Osvaldo Phillip:101 Country Rd.:999-888-7777:o_phil@aol.com
---
What would you like to update about this record?
Enter up to 4 choices with no spaces (e.g. bcd)

 (a) Name
 (b) Address
 (c) Phone Number
 (d) Email
 (e) Nothing

Choice: bc
Input updated Address: 900 Mansion Ln.
Input updated Phone Number: 214-315-9922
Updating existing record.


Welcome to Contact Database!
**************************
 (a) Add Record
 (b) Search record
 (c) Update record
 (d) Remove Record
 (e) Display Entire Database (Admin Only)
 (f) Exit

Enter choice: e
Enter password:
Showing database..

Osvaldo Phillip  100 Country Rd.  333-444-5555  olly@gmail.com
Osvaldo Phillip  900 Mansion Ln.  214-315-9922  o_phil@aol.com
```

Section IV - Implementation

```
#!/bin/bash
# Record Searching

            #Declaring the field variables with delimiters for prompting below.
NAME=":"
ADDRESS=":"
PHONE_NUMBER=":"
EMAIL=":"
line1=":"
                              #Declaring the variables used for input validation
fields="x"
input=""
minEmailCount=5
minPhoneDigits=7
until [[ $fields =~ ^[a-d]*$ ]]                          #Validating user's
choice ensuring the their query only contains a, b, c, and/or d.
do
clear
echo -e "Which contact fields would you like to search?\nEnter up to 4 choices
with no spaces (e.g. abd) \n";
echo -e " (a) Full Name \n (b) Address \n (c) Phone Number \n (d) E-Mail \n";
                     #Prompting the user and getting user's choice in next line
read -p "Enter choice: " fields
done

for (( f=0; f<${#fields}; f++));
do
#choice=$f
choice=${fields:$f:1}
case $choice in
"a"|"A"|"1")                                        #If user entered 'a' then get
the name to be searched
while [[ $NAME == *":"* ]]
Do                                                  #while the name contains the
delimiter continue to prompt for the name
read -p "Search for contact name: " NAME
done;;
"b"|"B"|"2")                                        #If user entered 'b' then get
the address to be searched
while [[ $ADDRESS == *":"* ]]
Do                                                  #while the address contains the
delimiter, continue to prompt for the address
read -p "Search for contact address: " ADDRESS
done;;
```

```
"c"|"C"|"3”)                                    #If user entered ‘c’ then get
the phone number to be searched

while [[ $PHONE_NUMBER == *":"* ]]  || [[ ${#PHONE_NUMBER} -lt $minPhoneDigits
]]
Do                                              #while the phone number
contains the delimiter, continue to prompt for the phone number
read -p "Search for contact phone number (at least ${minPhoneDigits} digits): "
PHONE_NUMBER
done;;

"d"|"D"|"4”                                     #If user entered ‘d’ then get the
email address to be searched

while [[ $EMAIL == *":"* ]] || [[ ${#EMAIL} -lt $minEmailCount ]]
do                                              #while the email contains the
delimiter continue to prompt for the email
read -p "Search for contact email: " EMAIL
done;;
esac
done
#get all lines from database.txt that include name, address, phone number, or
email and save them in contactInfo
contactInfo=$(grep -i "$NAME" database.txt | grep -i "$ADDRESS" | grep -i
"$PHONE_NUMBER" | grep -i "$EMAIL"
if [[ ${#contactInfo} != 0 ]]                   #checking that contactInfo is
not empty
then

OIFS=$IFS                                       #Saving the internal field
separator (IFS)
IFS="        #Setting the IFS to newline
"
count2=0                                        #creating a count variable.
for this in $contactInfo                        #for loop to navigate through
the results from grep
do
IFS=":"                                         #setting IFS to “:” to separate
the fields in the result line
read -ra x <<< "${this}"                        #reads all the fields in the
line separated by “:” into tan array (x is an array)

#the grep command gets any line that contains the string searched for in any
field. The following if statement makes sure that if the name is searched then
only those lines that have the matching string in the name field are shown
(the name could appear in the address, email etc.). Same comparisons are made
for address, phone number, and email. It makes sure that either the field is
```

```
":" which means that the user never entered the value for it, or the particular field for which the user entered the value matches the results. The ${x,,} formatting is used to make the comparison case insensitive.
if ([ "$NAME" == ":" ] || [ "${NAME,,}" == "${x[count2],,}" ]) && ([ "$ADDRESS" == ":" ] || [ "${ADDRESS,,}" == "${x[count2+1],,}" ]) && ([ "$PHONE_NUMBER" == ":" ] || [ "$PHONE_NUMBER" == "${x[count2+2]}" ]) && ([ "$EMAIL" == ":" ] || [ "${EMAIL,,}" == "${x[count2+3],,}" ])
then                                                    #if it is a match then print the contact information stored in array x
echo -e "----------------------------\nContact Information\n"
echo "Name: ${x[count2]}"
echo "Address: ${x[count2+1]}"
echo "Phone Number: ${x[count2+2]}"
echo "Email: ${x[count2+3]}"
echo -e "----------------------------\n"
else                                                    #if there was no match then it tell the user contact not found.
echo -e "Contact not found. Please enter full name and/or address.\n"
fi
IFS="    #sets IFS back to newline
"
done
IFS=$OIFS                                               #sets IFS back to its default value

else                                                    #if contactInformation variable was empty then it prints an error message
echo -e "Error: One or more fields did not exist in the database.\nTry simplifying your search.\n";
fi
```

Section IV (2)

(a) Only one exactly matching record query

```
Which contact fields would you like to search?
Enter up to 4 choices with no spaces (e.g. abd)

 (a) Full Name
 (b) Address
 (c) Phone Number
 (d) E-Mail

Enter choice: a
Search for contact name: Barack Obama
-----------------------------
Contact Information

Name: Barack Obama
Address: 440 Winchester Ave
Phone Number: 300-989-8299
Email: barry_o@hotmail.com
-----------------------------
```

One record with more specific query:

```
Which contact fields would you like to search?
Enter up to 4 choices with no spaces (e.g. abd)

 (a) Full Name
 (b) Address
 (c) Phone Number
 (d) E-Mail

Enter choice: ad
Search for contact name: Osvaldo Phillip
Search for contact email: o_phil@aol.com
-----------------------------
Contact Information

Name: Osvaldo Phillip
Address: 900 Mansion Ln.
Phone Number: 214-315-9922
Email: o_phil@aol.com
-----------------------------
```

(b) Multiple matching records for a query

```
Which contact fields would you like to search?
Enter up to 4 choices with no spaces (e.g. abd)

 (a) Full Name
 (b) Address
 (c) Phone Number
 (d) E-Mail

Enter choice: a
Search for contact name: Osvaldo Phillip
----------------------------
Contact Information

Name: Osvaldo Phillip
Address: 100 Country Rd.
Phone Number: 333-444-5555
Email: olly@gmail.com
----------------------------

----------------------------
Contact Information

Name: Osvaldo Phillip
Address: 900 Mansion Ln.
Phone Number: 214-315-9922
Email: o_phil@aol.com
----------------------------
```

(c) No matching query:

```
Which contact fields would you like to search?
Enter up to 4 choices with no spaces (e.g. abd)

 (a) Full Name
 (b) Address
 (c) Phone Number
 (d) E-Mail

Enter choice: a
Search for contact name: NotaReal Contact
Error: One or more fields did not exist in the database.
Try simplying your search.
```