ICT159 Assignment 2 (*ICT283 Revision: Ignore marks breakdown*)

Kim Andrew Dela Cruz

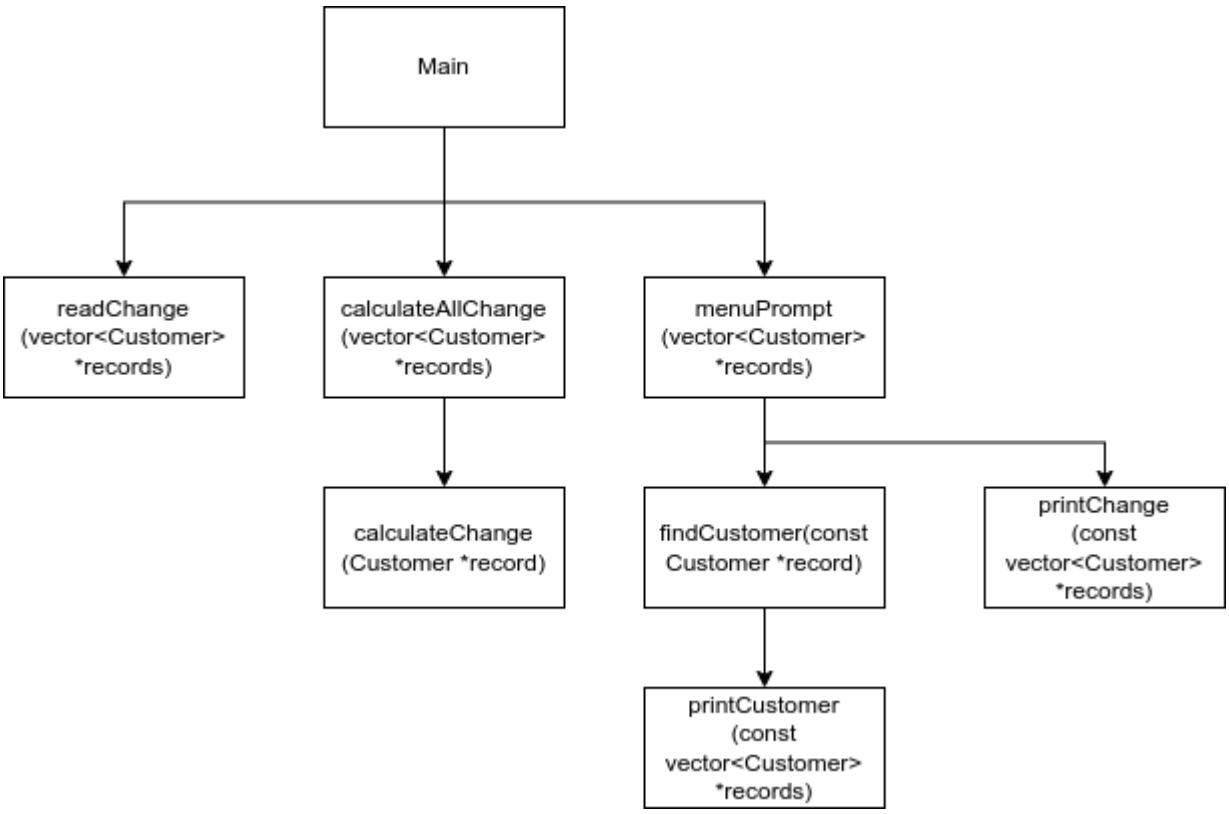
35282436

1. Assumptions (5%)

Assumptions:

- User knows that a change.csv file is generated/updated after program exit.
- User does not make any typos and follows the correct sentence structure when adding a new customer in the coins.txt file.
- User does not add any newline, whitespace, tab line in the coins.txt file.
- User knows that the find function for the program is case sensitive.

2. Structure Chart (5%)



3. Algorithm (20%)

```
STRUCT CUSTOMER
String name
int coinAmount
String currency
int coinDenoCount1
```

```
int coinDenoCount2
int coinDenoCount3
int coinDenoCount4
END STRUCT

MAIN
Array<Customer> records = new Array <Customer>
readChange(record)
calculateChange(records)

WHILE(ExitLoop==False)

PRINT(//MENU PROMPT...)
SCAN(String choice)

SWITCH(choice)
CASE 1:
findCustomer(Array)
CASE 2:
printChange
ExitLoop=True
BREAK
END SWITCH

END WHILE LOOP

END MAIN


readChange(Array<Customer> record)
READ FILE("coins.txt")

WHILE(!EOF) //End of file
WHILE(!EOL) //End of line
String name
String change
String currency
String skip

// Example line:
// Jane 30 cents in AU$

name = first word
change = second word
skip = third word
skip = fourth word
currency = fifth word


record.name(name)
```

```
record.change(change)
record.currency(currency)
END WHILE
END WHILE
END readChange

calculateChange(Array<Customer> record)
Int total

IF(record.currency == "//CurrencyCode(ex. US$)")
WHILE(total != 0)
IF(total >= 50)
total -=50
record.coinDenoCount1++
IF(total....
.... // Repeat for all coin Denominations
END WHILE
END IF

IF(record.currenc.....
......
...... // Repeat for other currencies

END calculateChange

findCustomer(Array<Customer> record)
    DISPLAY "Name: "
INPUT customerName
BOOL found = FALSE

FOR EACH customer IN records
          IF (customerName == customer.name)
SET found = TRUE
printCustomerDetails(customer)
          ENDIF
     ENDFOR

     IF NOT found THEN
        DISPLAY "Not found"
     END IF

END FindCustomer


printCustomerDetails(Customer customer)
PRINT ("Customer: " + customer.name + " " + customer.coinAmount +
" cents in" +  customer.currency + " Change: ")

IF(record.currency = "//currency code (Ex. US$)")
IF (record.coinDenoCount1 != 0)
```

```
PRINT ("50 Cents: " + record.coinDenoCount1)
IF (record.coinDen... //Repeat for all coin denominations
IF(record.curr...
      ...
      ... //Repeat for other currencies
END PrintCustomerDetails

printChange(Array<Customer> records)
OUTPUT FILE("change.csv")

FOR(int i = 0; i < records.size ;i++)
PRINT (records.name + ","
records.coinAmount + " cents in "
records.currency + " is "
records.coinDenoCount1 + ","
records.coinDenoCount2 + ","
records.coinDenoCount3 + ","
records.coinDenoCount4)
END FOR
```

4. Test Table (10%)

A set of test data in tabular form with expected results and desk check results from your algorithm. Each test data must be justified – reason for selecting that data. No marks will be awarded unless justification for each test data is provided.

Add rows to the following table as needed. Table can span more than one page. Each test id tests only one condition for the desk check.

For this assignment, there can be up to 10 records in a data file. In the test table below, you might have one test id for 10 records. So the actual 10 records must be in one cell of the test table in the column *Actual data*. Of course there are other test conditions and you need to include those too.

Faking the outcome of any test will result in no marks given for this entire section. What that means is that if you have a few hundred tests which are fine, but you faked/falsified the outcome of just one, you will get a mark of 0.

Test id	**Test description/justification – what is the test for and why this particular test.**	**Actual data for this test**	**Expected output**	**Actual desk check result when desk check is carried out**	**Desk check outcome – Pass/Fail**
1	Correct customer result when selecting "1" option of the menu.	Customer Jane	Should display 2 Customer Jane, one for AUS currency and US for the other	Program displays 2 Customer Jane, one for AUS currency and US for the other.	Pass
2	Verify if customer change calculation is correct	Customer Joe	85 cents EUR: 20=4 10=0 5=1 1=0	20 x 4 = 80 10 x 0 = 0 5 x 1 = 5 1 x 0 = 0 80+0+5+0 = 85	Pass
3	Check if Customers change results are copied into a file named "change.csv" upon exit	All Customers	Program is Exited, change.csv is created/updated, file is not	Upon entering "2" in the menu prompt, the program exits, and a new file is created/updated, and	Pass

			empty.	when opened via text editor, displays all customer and their change.	

5. Code (50%)

Name and purpose of functions/modules in the source code files. Do not put actual source code here. Code exists as separate source code files that are submitted. Source code files (.c and .h) must be submitted separately and the source code must build (compile and link) to create an executable that operates correctly. Make sure you use the code style required in the unit. No marks awarded if the source code does not build and run.

Extend the following table as needed. Functions/modules need to match what is in the structure chart. If it is the same file name for a number of functions/modules, you write the file name once in the *File name* column for the first function/module listed in the table.

File name	**Name of Functions/modules in the file**	**Purpose of the Function/module**
main.cpp	main()	Calls every other function.
	menuPrompt()	Display menu and handle user input.
	unitTest()	Tests and displays all customer records inside the Customer Vector Struct.
FileHandler.cpp	readChange()	Opens the .txt file which contains the customer data and reads each line and inserts the data into the Customer Vector Struct accordingly.
	printChange()	Creates a file called “change.csv”, inserts all records inside the Customer Vector Struct
Customer.cpp	printCustomer()	Designed as a helper function for findCustomer() but can be used independently, this function prints the change information of the customer.
	findCustomer()	Handles user input, searches the name that the user has entered inside the Customer Vector Struct.
Change.cpp	calculateChange()	Designed as a helper function for calculateAllChange() but can be used independently, this function calculates the customer’s change amount into coin denominations and inserts the coin denomination count data into the customer struct.
	calculateAllChange()	Calculates the coin denomination count for all customers inside the Customer Vector Struct.

6. Results of Program Testing (5%)

Test id	Test description/justification – what is the test for and why this particular test.	Actual data for this test	Expected output	Actual program output when test is carried out	Test run outcome – Pass/Fail
1	Correct customer result when selecting "1" option of the menu.	Customer Jane	Should display 2 Customer Jane, one for AUS currency and US for the other	Figure 1.	Pass
2	Verify if customer change calculation is correct	Customer Joe	85 cents EUR: 20=4 10=0 5=1 1=0	Figure 2.	Pass
3	Check if Customers change results are copied into a file named "change.csv" upon exit	All Customers	Program is Exited, change.csv is created/updated, file is not empty.	Figure 3.	Pass

Figure 1.
Test ID 1

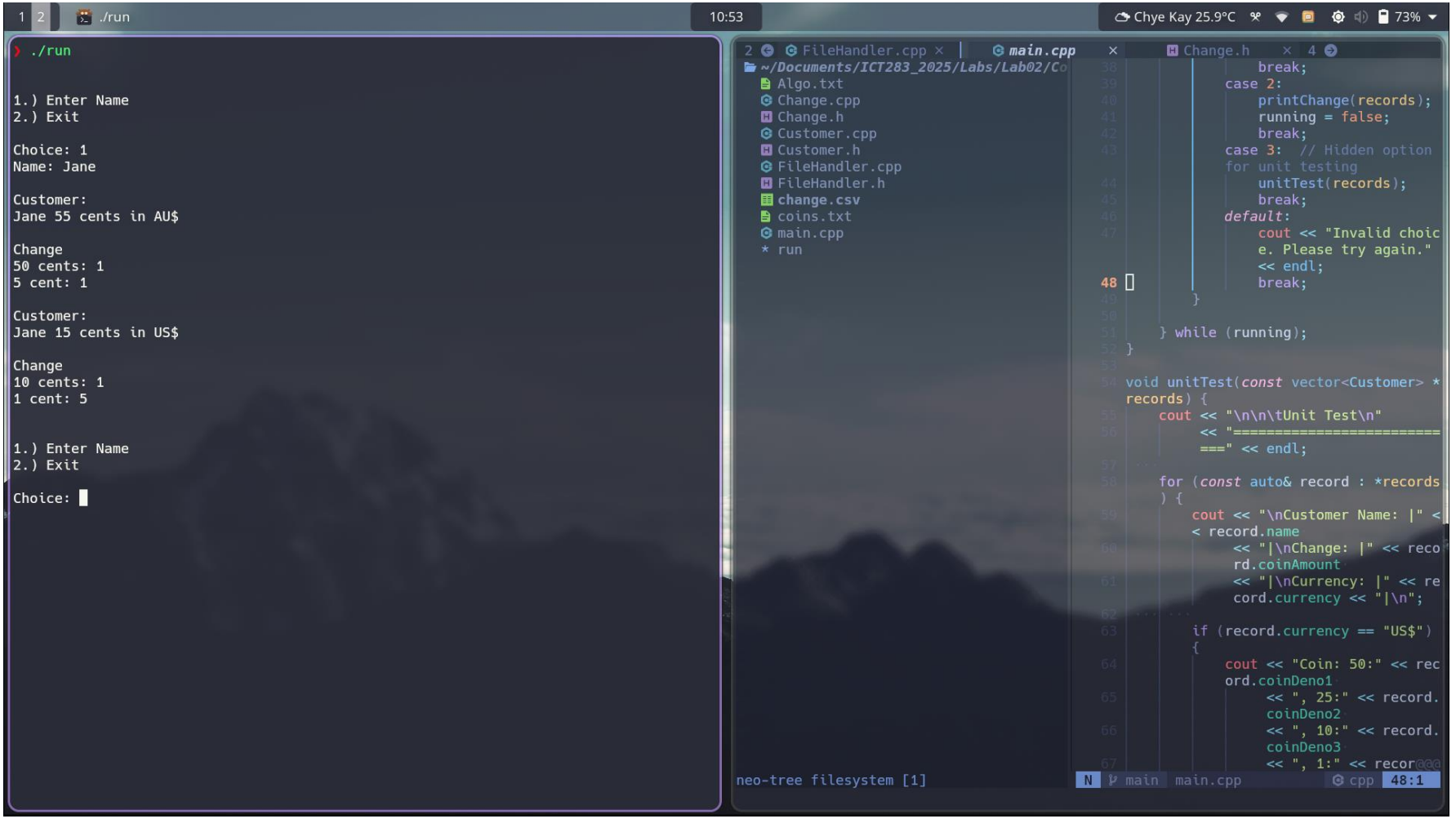


Figure 2.
Test ID 2

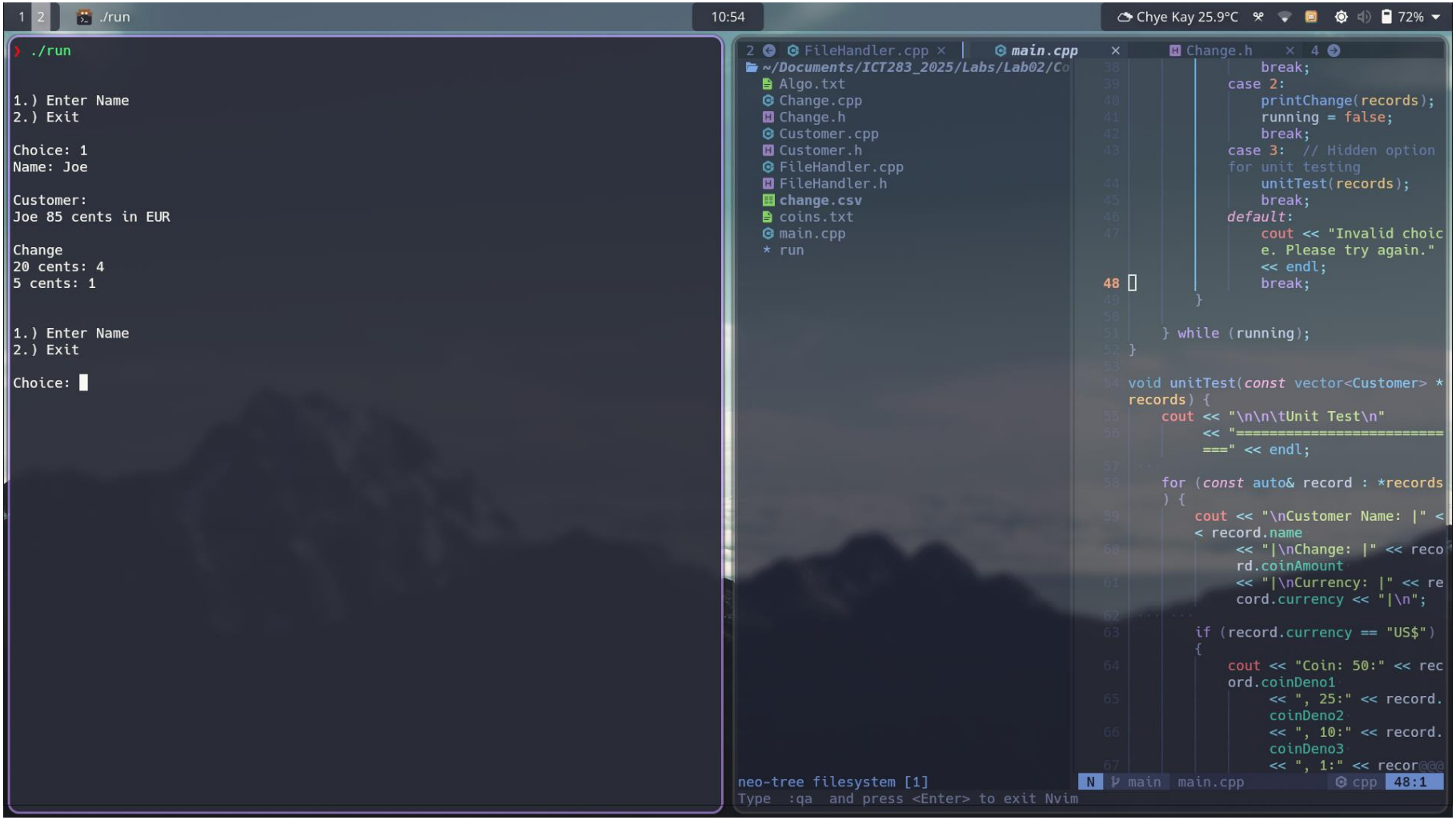
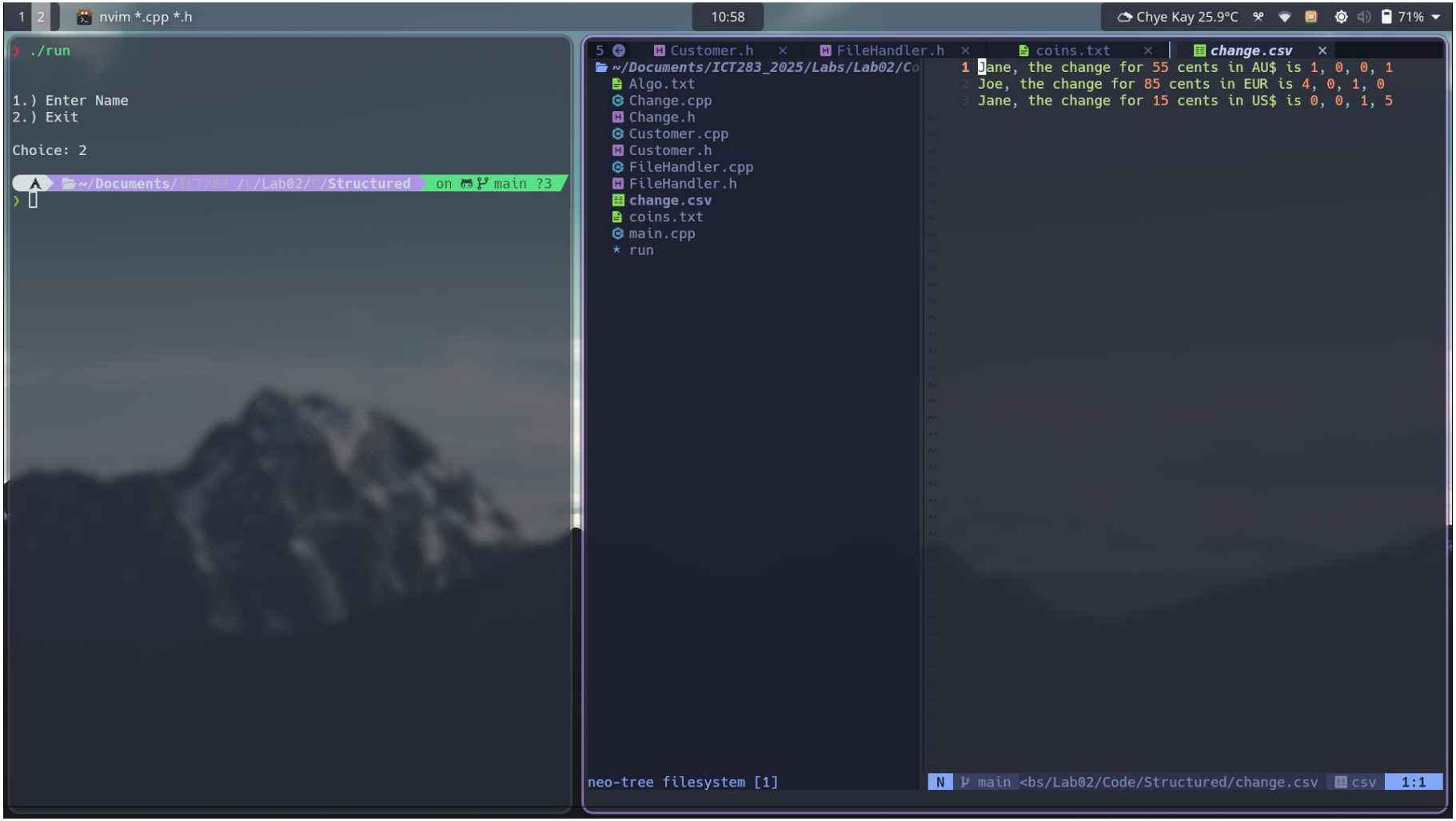


Figure 3.
Test ID 3


1 2
nvim *.cpp *.h
10:58
Chye Kay 25.9°C
71%
./run
1.) Enter Name
2.) Exit
Choice: 2
~/Documents/ICT283_2025/Labs/Lab02/Co/Structured on main ?3
Customer.h
FileHandler.h
coins.txt
change.csv
~/Documents/ICT283_2025/Labs/Lab02/Co
Algo.txt
Change.cpp
Change.h
Customer.cpp
Customer.h
FileHandler.cpp
FileHandler.h
change.csv
coins.txt
main.cpp
run
1 Jane, the change for 55 cents in AU$ is 1, 0, 0, 1
2 Joe, the change for 85 cents in EUR is 4, 0, 1, 0
3 Jane, the change for 15 cents in US$ is 0, 0, 1, 5
neo-tree filesystem [1]
N main <bs/Lab02/Code/Structured/change.csv csv 1:1

7. Self-Assessment (5%)

My program accomplishes all the requirements and tasks that is asked for, functionality wise. If I were to improve the solution, I would make it so that the program prompts the user for a currency after typing the customer the user wants to find. so that multiple records with the same name will not display all together making it more readable and neater. A problem that I encountered was the getline() function reading the newline after each line, I resolved this problem in two ways, first solution is for my C program version where I used the strok() function, I made a function to remove whitelines, newlines and tablines in a string and called this function before I inserted it into the struct. My second solution was to use the stringstream library and call istringstream() function instead of strok() (this was when I was making this C++ version of the program which is the current program).