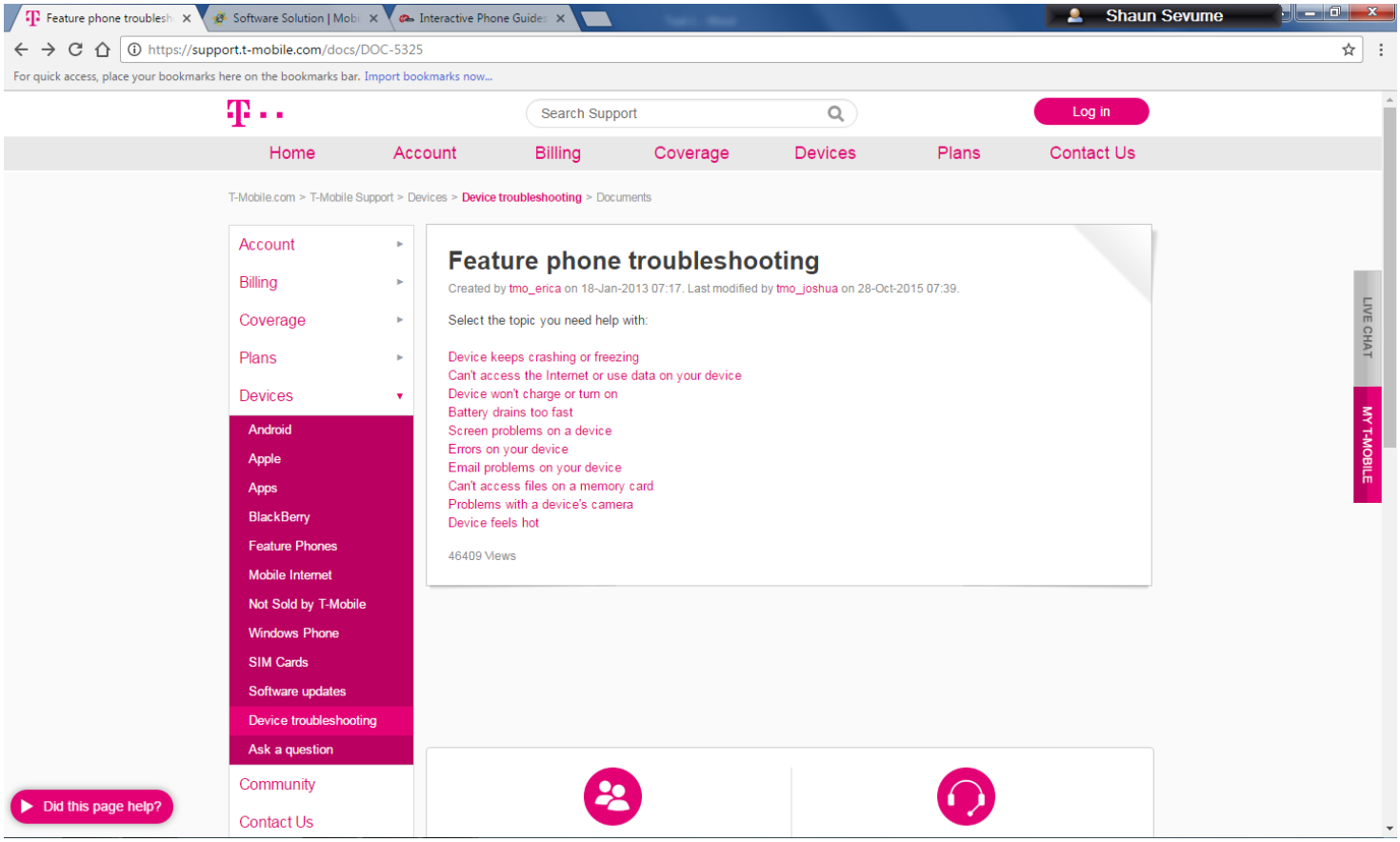
1. Analysis

I'm required to create a troubleshooting program for mobile phones.

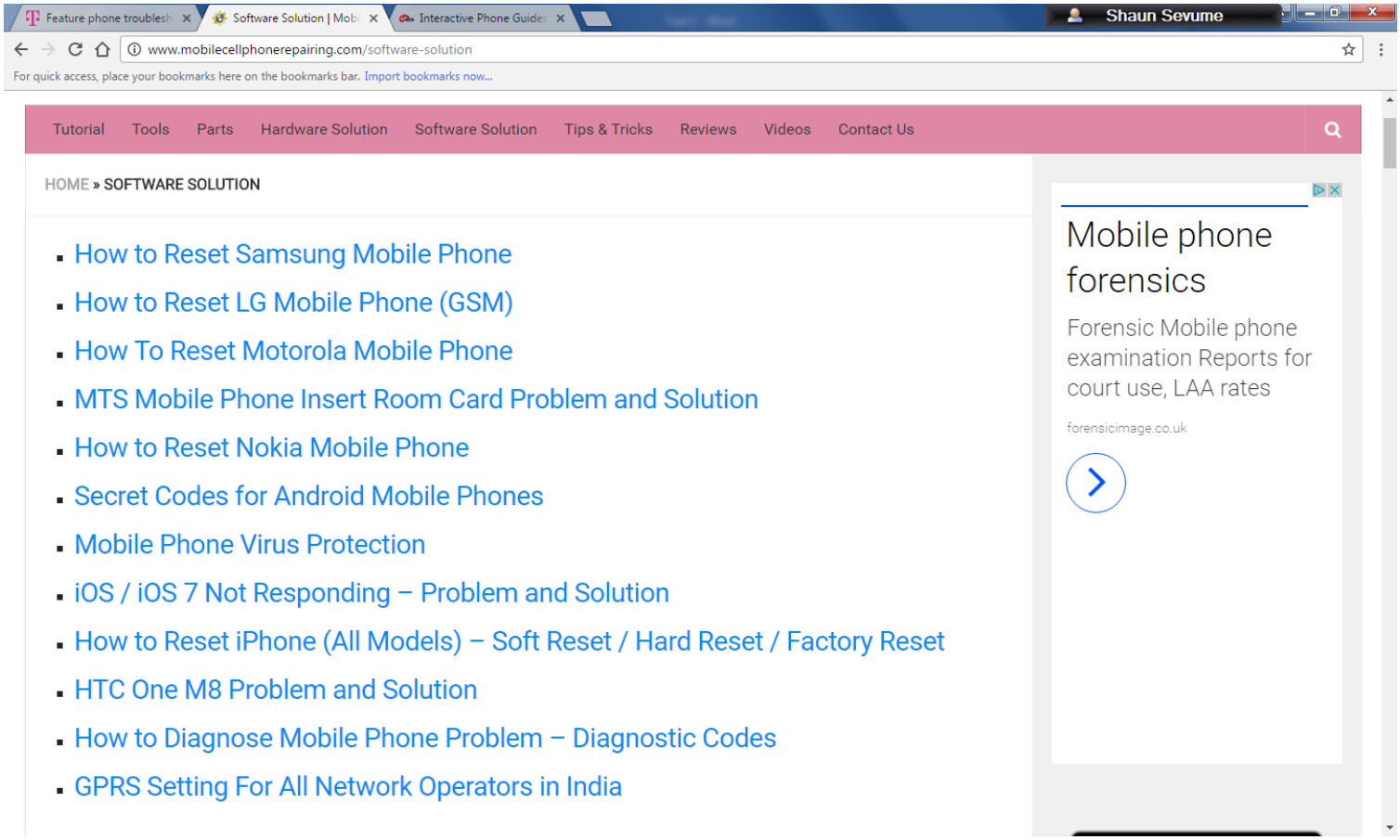
I did some research onto the current mobile troubleshooting programs online, and I found three, all of which were designed in a very similar fashion. In all of them, they required the user to click around to find what they were looking for.

The first website was T-Mobile, a popular mobile network. https://support.t-mobile.com/docs/DOC-5325



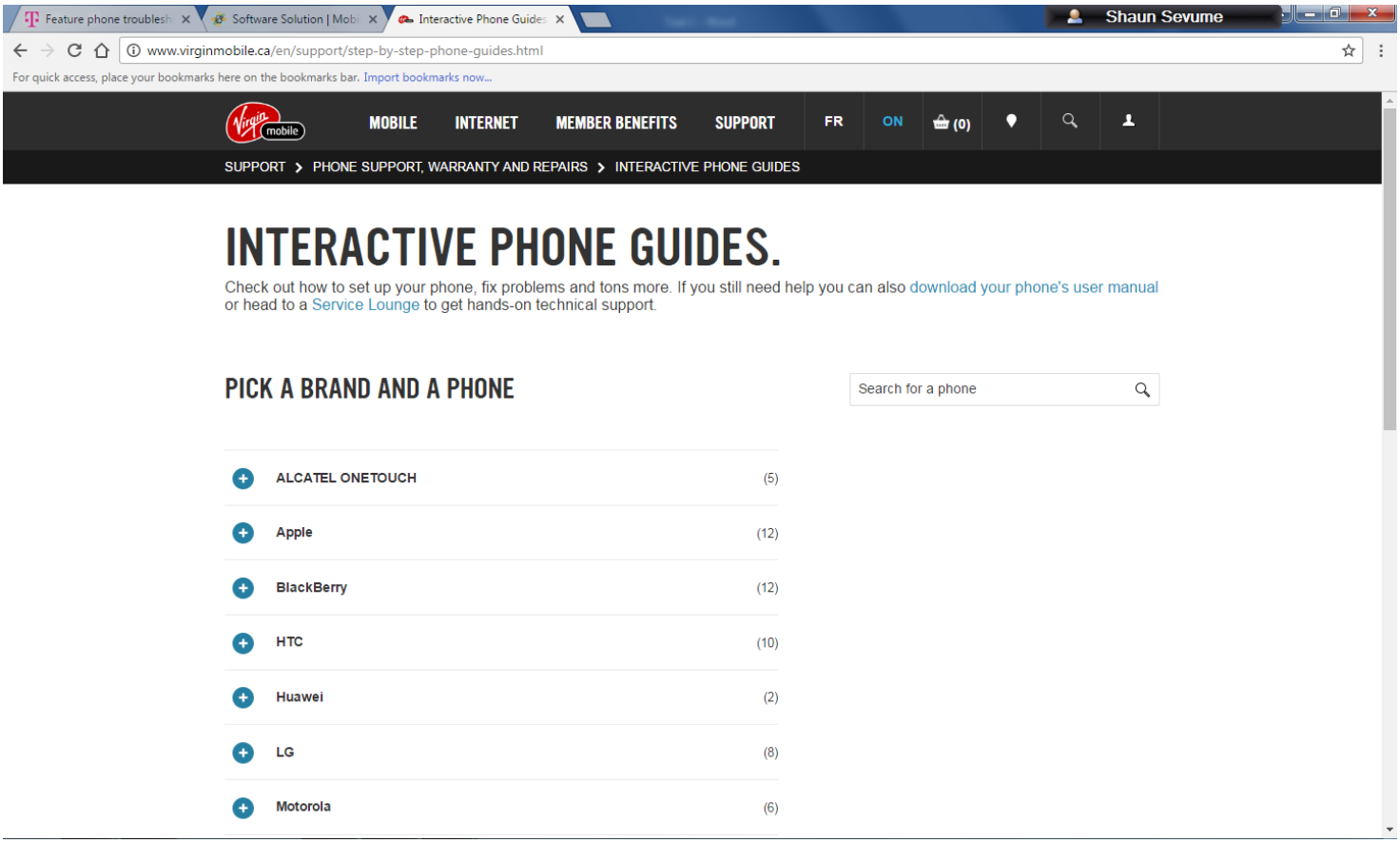
This site is a good example of what my program should be like. It has clear, broad problems and solutions. There is even an "Ask a question" tab and "Contact us", so T-Mobile are there to support you if you don't find your answer. Though not every problem is covered, it's easy to add to the list. There's also a search at the top to get around the site quickly. The only downside was the fact that solutions were not device specific, so instructions would be vague examples rather than clear direct ones for the user's particular device/OS.

The second website is Mobile phone cell repairing. It seemed to be a simple mobile troubleshooting site. http://www.mobilecellphonerepairing.com/software-solution



Similar to the T-Mobile website, this website had clear problems and solutions, and some were even device specific. A search bar and a “Contact us” tab also featured here, but at times user friendliness felt limited, and a lot of reading had to be done before making choices.

The third website was Virgin Mobile, another popular network provider.
http://www.virginmobile.ca/en/support/step-by-step-phone-guides.html



This particular website required you to scroll down until you found your device model, and then click it, which would present a list of options to choose from, which the user of course had to click on. You could search for the make and model, but not all instances were there present. I don't find this very user-friendly, because it requires a lot of input from the user, and they have to find their way around the site to get their answer. My program is going to be the opposite; it will do all the work for the user.

One thing to note is that none of these are actually *programs* but rather a collection of hyperlinks on a site that contain information on specific subjects.

My program will be user friendly, easily understandable and easy to use, requiring little user input to run. It will be targeted at people of a 12+ age and will be able to run on most PC's, as it isn't a resource consuming program. Minimum requirements will be an Intel Pentium processor @500MHz and 256MB of RAM and an operating system of Windows XP. Other components needed include a mouse, a monitor and a keyboard as well as additional components found on a motherboard, such as a hard drive. My program must be able ask the user for their name, and using a range of questions, help them diagnose what's wrong with their phone. If the problem cannot be solved, then the program will give you further advice which may be along the lines of "contact your supplier regarding (insert issue here)". The simplicity yet effectiveness of my program will be what makes it stand out from the others. Rather than being scroll and click based, it will be typing based, which is in my opinion easier. It will also be available to download, so you won't need a constant internet connection to access it.

Success criteria

- Greet the user with their name
- Ask them a series of questions with different outcomes
- Using the answers given, try to diagnose what is wrong with the phone.
- Give an appropriate solution and if no solution can be found, give further advice.
- Based on the input, give an appropriate output
- Maintain a sense of user-friendliness so that it stands out from other programs.
- Solve a phone not being able to turn on
- Solve a phone not being able to charge
- Solve a phone's touch screen problems
- Solve a phone's volume problems
- Solve phone calling issues
- Solve texting issues
- Solve phones unable to connect to the internet
- Solve phone brightness problems
- Solve phone microphone problems

Test plan

Test no.	Testing	Input (if applicable)	Expected output
1	Does it address the user by their name based on their input?	Shaun	Okay Shaun…
2	Does it wait the required time before carrying out the next function using time.sleep?		(After the amount of time specified in the time.sleep function) Okay Shaun…
3	Does it recognize inputs in both lower and upper case when	n/N	Have a nice day!

	asking to return to the main menu?		
4	Does it crash when something unexpected is inputted?	qwerty (anything random)	Error
5	Is my code correctly indented?	N/A	(Indentation) Error
6	Are variables correctly defined?	N/A	(Unidentified variable) Error
7	Is there an else statement in case the if statements are not met?	N/A	If you don't put else, there will be no output if no conditions are met.
8	Does it give further advice if issue wasn't solved?	(Did this solve your issue?) N	(Give more advice) … contact your phone manufacturer regarding your issue instead.
9	Does it give an appropriate output based on the input?	(Did this solve your issue?) Y	We're glad to have solved your issue!
10	Does it keep on asking for an input for a specific variable before a condition is met? (Iteration)	(Anything that isn't y,Y/n,N)	Please enter y or n. Return to main menu?

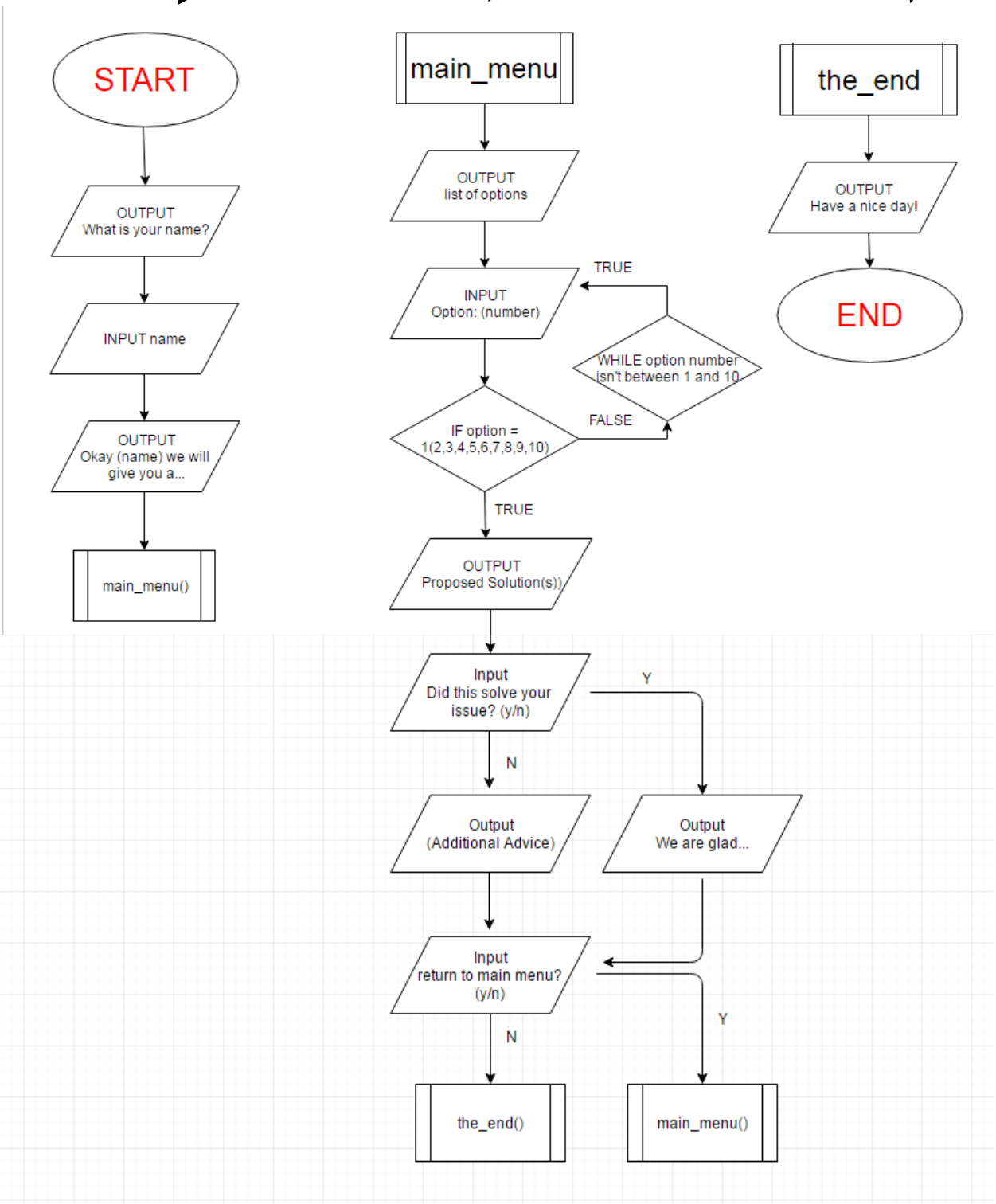
2. Design

Flowchart

This flowchart is only used once when the program is first opened. It is here the variable "name" is given a value.

This flowchart represents the main structure of the code. It is a generalisation of all of the options, since they all go along roughly the same lines. Where there is an ellipsis represents more text, since it obviously wouldn't fit in the box.

This flowchart is used to end the program should the user wish to do so. It simply tells them to have a nice day before exiting.



Pseudocode

```
START

IMPORT time

OUTPUT (Welcome message)

INPUT Name


FUNCTION Main menu:
                    OUTPUT Options
                    INPUT option number
                    IF option number = (1-10) THEN
                         OUTPUT Solution (Based on option number)
                    ELSE:
                         CALL Main Menu
                    END IF
                    CALL Feedback


FUNCTION Return to menu:
                    WHILE return ISN'T y, OR n:
                         OUTPUT "Return to main menu?"
                         INPUT return
                         LOWERCASE return
                         IF return = y THEN
                                 CALL Main Menu
                         ELSE IF return = n THEN
                                 CALL The End
                         END IF
                    END WHILE


FUNCTION The end:
                    OUTPUT "Have a nice day"
                    END
```

Variable and Validation table

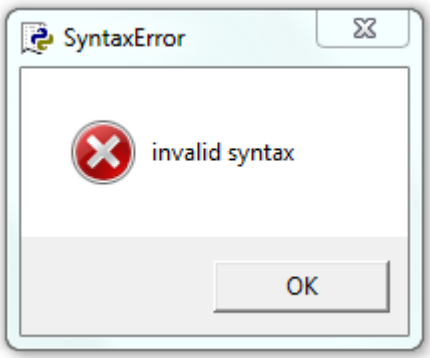
Variable Name	Type	Validation	Description
time	Integer	All numbers above 0	Allows the use of the time.sleep function which will decide how long the program waits before executing the next line.
name	String	Any character	String value inputted by the user is assigned to the variable name. The user may wish to input numbers or other characters such as letters with accents, and therefore the variable is not just limited to a-z, A-Z for user friendliness.
main_menu	Function	N/A	This has no value, but is used for the program to return to the start if the user wishes to do so based on their input for the return1 variable. (See return1 for more details.)
option	String	Numbers without a decimal point in the range 1-10	String value inputted by the user is assigned to the variable: option. If the string value is between the numbers 1 and 10, the program will proceed to execute one of the if statements, based on the user's input, which means they will get the help that's most appropriate to their situation. Since the if statements only work with numbers 1-10, any other input will result in the program prompting you to input something else, until it satisfies one of the conditions for the if statements.
return1	String	Characters y/n, Y/N	String value inputted by the user is assigned to the variable: return1. If the input is y or Y, then it will return to the main menu of the code by calling main_menu(). However, if the input is n or N, then it will simply tell the user to have a nice day before stopping.
feedback	String	Characters y/Y or n/N	The user is asked if their problem has been resolved beforehand. The answer is then assigned to the variable feedback and an appropriate output will be displayed depending on if the user inputted y or n.
feedback	Function	After the user inputs an option number and the program outputs a solution.	The user is asked if their problem has been resolved beforehand. Depending on the input, they may be taken to the return_toMenu function or given further advice first.
return_toMenu	Function	After the user has been given extra advice or if they picked option 10	Contains a while loop that keeps asking the user if they want to return to the menu until they enter y or n.

the_end	Function	If the user inputs n when asked if they want to return to the menu	This is the function that terminates the program, using the SystemExit command.

3. Development

After printing all the possible choices, the users could pick, I had to declare a variable that would allow them to pick an option, and based on their input, I would use an IF statement to give them the appropriate output. However, I kept on getting an error message when trying to declare a variable.

```
option = int(input("Option: "))
```



I re-read the code and identified my mistake.

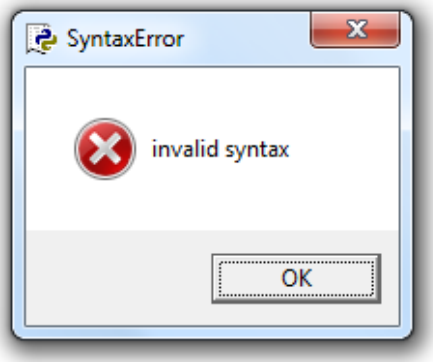
```
print("1.Your phone won't turn on")
print("2.Your phone won't charge")
print("3.Your phone's touch screen/keypad won't work")
print("4.Your phone does not play any sound. (This can include music or ringtones)")
print("5.Your phone is unable to place calls.")
print("6.Your phone is unable to send text messages")
print("7.Your phone is unable to connect to the internet")
print("8.Your phone's display is dim.")
print("9.Your phone's microphone isn't working properly")
print("10.Your problem is not listed above." #<--THERE WAS NO BRACKET HERE!!!
```

Fixed Code:

```
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen/keypad won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.
Option:
```

When trying to make an if statement, I got the invalid syntax message along with a highlighted colon. Also, the print command was being indented incorrectly. The syntax is a set of rules that defines how the python language is run. If it is 'invalid', then one of these rules are broken, therefore the code cannot be read and the program cannot run.

```
option = int(input("Option: ")

if option == "1":
                print("hi")
```



Same mistake. It should've been option = int(input("Option: ")). And also I didn't need to add int before input anyway, so I removed it. What was invalid here was the missing bracket, so the code couldn't be read.

```
option = int(input("Option: ") #<-- ONE BRACKET WAS MISSING HERE!!!!

if option == "1":
            print("hi")
```

Fixed Code:

```
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen/keypad won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.
Option: 1
hi
```

I noticed that if the user entered anything that wasn't between 1 and 10, they got this in return:

```
Option: twenty
None
>>>
```

So to fix it, I added an else statement near the end of the code, which meant that if would ask them to enter a valid number and give them the options list again, as well as the line to input their number again. The whole code was based on if statements (if option == 1 print… etc) so adding an else statement at the end worked because it would mean that if they entered anything for the options variable that wasn't a number between 1 and 10, they would be prompted to try again.

```
else:
    print("Please enter a valid number")
    print(main_menu())
```

```
Option: twenty
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.
Option: #(list printed again)
```

When I first finished my code, I realized that it just ended once a solution had been reached, which I didn't like, because if they wanted to solve another problem, they had to restart the program. To solve this problem, I used the def function, entering def main_menu(): at the start of the listing of the options so that every time the program was told to print(main_menu()), it took you back to the options list, where you could select another one and get another solution. However, I kept getting an indentation error, so I had to indent all code after it in the def main_menu(): so that it basically restarted the code for the user, if they chose to. However, the list was no longer printing the first time. It was then that I realized that I had only defined the main menu, and now I needed to tell python to actually print it. So right at the bottom, after the indented code, I put print(main_menu()) so that it printed. Now whenever the user entered 'y' when asked the question "Return to main menu?" the list was printed again for them to make another selection.

```
def main_menu(): #This is the main menu, where the user will be able to select their opti
    print("1.Your phone won't turn on")#These will first appear to the user and they will
    print("2.Your phone won't charge")
    print("3.Your phone's touch screen won't work")
    print("4.Your phone does not play any sound. (This can include music or ringtones)")
    print("5.Your phone is unable to place calls.")
    print("6.Your phone is unable to send text messages")
    print("7.Your phone is unable to connect to the internet")
    print("8.Your phone's display is dim.")
    print("9.Your phone's microphone isn't working properly")
    print("10.Your problem is not listed above.")
```

The def main_menu() at the start was added so every time it was told to print it, it re-printed the list. All code after this was indented in the function.

```
print(main_menu())
```

Unindented at the end of the code so the options list printed the first time round.

```
Return to main menu?(y/n)y
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.
Option:
```

Options list re-printed if the user wishes to return to the main menu.

When the program asked the user if they wanted to return to the main menu, if they typed “y” or “Y”, it worked fine, but when they typed “n” or “N”, it did print “Have a nice day”, but would then ask if they wanted to return to the main menu again, stuck in a while loop.

```
return1 = 0

while return1 != "y" or return1 != "Y" or return1 != "n" or return1 != "N":

    return1 = input("Return to main menu?(y/n)")

    if return1 == "y" or return1 == "Y":
        print(main_menu())

    elif return1 == "n" or return1 == "N":
We are glad to have helped you solve your issue,Shaun!
Return to main menu?(y/n)n
Have a nice day!
Return to main menu?(y/n)n
Have a nice day!
Return to main menu?(y/n)n
Have a nice day!
Return to main menu?(y/n)
```

I defined a function called the_end() but it didn’t work. It was still repeating the question.

```
def the_end():
    print("Have a nice day!")
```

Then, after doing some research (http://stackoverflow.com/questions/19747371/python-exit-commands-why-so-many-and-when-should-each-be-used), I found a solution.

Or, even better in my opinion, you can just do directly what `sys.exit` does behind the scenes and run:

```
raise SystemExit
```

This way, you do not need to import `sys` first.

However, this choice is simply one on style and is purely up to you.

Using this solution, I adapted it to my own code. I learnt that by entering raise SystemExit, the program would execute this line and then exit, breaking the while loop and ending, so the question wouldn’t be repeated. It was also better than using sys.exit() because it terminated the program, so that no more lines were executed, instead of just bringing up a confirmation window to actually exit python shell.

Fixed Code:

```
def the_end():
    print("Have a nice day!")
    raise SystemExit
```

```
Return to main menu?(y/n)n
Have a nice day!
>>>
```

I also changed the while loop to be more efficient. Instead of specifying both lower and uppercase inputs, I used the .lower function, so that the input would be converted to lowercase regardless, so in the code, I would only need to specify inputs in lower case.

Before:

```
return1 = 0

while return1 != "y" or return1 != "Y" or return1 != "n" or return1 != "N":

    return1 = input("Return to main menu?(y/n)")

    if return1 == "y" or return1 == "Y":
        print(main_menu())

    elif return1 == "n" or return1 == "N":
        print("Have a nice day!")
```

After:

```
while return1 != "y" or return1 != "n": #In this while loop, it will

    return1 = input("\nReturn to main menu?(y/n)")#See line 18
    return1_lower = return1.lower()

    if return1_lower == "y":#See line 25
        main_menu()#Here, the main_menu() function is called.

    elif return1_lower == "n":#See line 24
        the_end()#Here, the the_end() function is called.
```

This didn't affect the output, but made the code look neater with less words and easier if in future I ever wanted to add any other possible responses, as I'd only have to add them for lowercase inputs.

When looking back over my code, I saw a way to improve it. The screenshot below shows how before; I had used multiple print functions to display the options in a list. This was due to the fact that at the time of making the code, I didn't know how to force text onto a new line, so my code looked like this:

```
def main_menu(): #This is the main menu, where the user will be able to select their option. T
    print("1.Your phone won't turn on")
    print("2.Your phone won't charge")
    print("3.Your phone's touch screen won't work")
    print("4.Your phone does not play any sound. (This can include music or ringtones)")
    print("5.Your phone is unable to place calls.")
    print("6.Your phone is unable to send text messages")
    print("7.Your phone is unable to connect to the internet")
    print("8.Your phone's display is dim.")
    print("9.Your phone's microphone isn't working properly")
    print("10.Your problem is not listed above.")
```

However, I learnt about the \n function sometime after, and I went back to implement it in the code. This is what the edited version looked like:

The same applied to all 10 options. They were all in the same print function, split with the \n function. This also makes it easier to add more options, instead of creating a whole new print function. I did the same thing for other parts of my code too.

Before:

```
elif option == "5":
    print("Sim Card: Make sure you have a SIM card inserted first! Unless you are attempting to call by internet, in which case, see option number 7. \n")
    print("Credit: Make sure you have sufficient credit if you are on a pay as-you-go plan or you have sufficient minitues if you are on a monthly plan.")
    print("Coverage: Make sure you have network coverage in your area. If not, try and move to an area with better coverage. If this still doesn't resolve your issue, talk t
    print("Valid Number: Make sure the number you want to call is valid and their phone is on. If not, a call will not take place.")
    time.sleep(5)#See lines 2 and 5
    feedback()

elif option == "6":
    print("Sim Card: Make sure you have a SIM card inserted first! Unless you are attempting to text by internet, in which case, see option number 7.")
    print("Credit: Make sure you have sufficient credit if you are on a pay as-you-go plan or you have sufficient texts if you are on a monthly plan.")
    print("Coverage: Make sure you have network coverage in your area. If not, try and move to an area with better coverage. ")
    print("Valid Number: Make sure the number you want to text is valid. If not, your text will not send.")
    time.sleep(5)#See lines 2 and 5
    feedback()

elif option == "7":
    print("Mobile data: If you're using your mobile data (3G/4G usually), ensure you have netowork coverage (Usually displayed in a top corner of your phone) and that you ha
    print("Wifi: Make sure that you are within a suitable range of a wifi hotspot or hub and that wifi is enabled in your settings. Make sure the souce of wifi is turned on
    time.sleep(5)#See lines 2 and 5
    feedback()
```

After:

```
elif option == "6":
    print("\nThis could mean a problem with your phone or network provider. Contact your network provider about the issue after making sure your phone isn't the pro

elif option == "7":
    print("\nMobile data: Contact your netowork provider about your issue. \nWifi: Contact your ISP about your issue.")

elif option == "8":
    print("\nThis could mean possible damage to your screen, possibly the light filters. This damage is repairable, but you will have to go to your phone's manufactu
```

Adding the \n to my code made it appear neater too. This is what it was like before:

```
---------- RESTART: N:\Task 1\Task 1 Code.py ----------
Welcome to the multicellular Troubeshooting program. If you have a problem with your phone, then you've come to the right place! Though we are only limited to a handful of solu
tions, we hope to expand and solve everyone's problems in the future.

To familiarize yourself with our program, we think it would be helpful if you enter your name.
Name: Shaun
Okay Shaun, we will give you a list of possible options. Remember, we are currently limited to only a few solutions, but hope to expand in the future. Please read the options b
elow and enter the appropriate number.
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.
Option: 4
Make sure you turn up your volume. This can be done by pressing the volume keys usually located on the side of your device. If not, try going to settings and looking for the op
tion relating to sound.
Did this solve your problem? (y/n)n
This could mean a problem with your speakers. Go to your phone's manufacturer or repair shop for further assistance.
Return to main menu?(y/n)n
Have a nice day
None
>>>
```

And this is what it looked like afterwards, when the \n function had been implemented across the code.

```
Welcome to the multicellular Troubeshooting program. If you have a problem with your phone, then you've come to the right place! Though we are only limited to a handful of
solutions, we hope to expand and solve everyone's problems in the future.

To familiarize yourself with our program, we think it would be helpful if you enter your name.
Name: Shaun

Okay Shaun, we will give you a list of possible options. Remember, we are currently limited to only a few solutions, but hope to expand in the future. Please read the options
below and enter the appropriate number.

1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.

Option: 2

Try using another cable and power adapter that you know works on another phone, to see if the problem lies with your phone or your charger. If your phone charges after this, th
en it was your charger that was the problem. If the phone isn't charging, this could mean there's something wrong with the charging port. check to see if there's any dust or th
ings obstructing it, and make sure the charger fits in properly(not hanging loosley) and your battery works too.

Did this solve your problem? (y/n)y
We are glad to have helped you solve your issue,Shaun!

Return to main menu?(y/n)n

Have a nice day!
>>>
```

Techniques Used

Technique/Keyword	What does it do?	How is it used in my code?
def	Used to define a function, or a piece of code. You can use this to split your code into blocks so that instead of repeating code over and over again, you simply call the appropriate function which will have the code written in. It means you only have to write the code once, and makes it easier to identify which part of your code has a problem (should any arise).	It is used to split the code into blocks, making it easier to identify problems and not needing to repeat code. For example, the the_end() function is used to exit the program. It only has to be called instead of being repeated.
Import	Enables the use of specific techniques, depending on what you choose to import	I used import time so I can use the time.sleep technique.
Print	Outputs whatever is specified in the speech marks and brackets.	I use the print function on various occasions, mostly to display instructions to the user.
Input	Assigns the user's input to a variable.	An example of when I have used this is when the user is asked their name. What they enter is assigned to the variable called name, made possible because of the input keyword.
While	Used to create a WHILE loop, where a piece of code is constantly repeated until the condition for the loop is either made true or false (depending on how you structured the loop.	I used a WHILE loop when the program was asking the user if their problem was solved or not. The condition was the input not being y or n, and whilst this is true, the while loop is run. Once the condition becomes false, the loop is broken, and the program can proceed.

Or	This is usually (but not limited to) with if statement. It gives python an alternative condition to satisfy that applies to the same variable.	I used it with the variable called feedback. The condition was whether the user's input was y OR n. The or function makes sure that both conditions are checked before the program carries on running.
If /elif	This is a statement that has a condition, which needs to be satisfied first. The condition could be a variable being equal to something, or the user's input matching a specified keyword. Once the condition is satisfied, the rest of the if statement is run. If the condition isn't satisfied, then it will usually run the next if statement (if there are any), run the else statement (if present) or just terminate if nothing is specified for what the program has to do should the statement not be satisfied.	An example of me using an if statement is when I put 'IF' option = 1. This means that if the variable called option has a value of 1 (based on user input), then whatever was specified to do if the condition was true is done. In this case, it prints information relative to phones not turning on, since option 1 was the phone not turning on.
Raise	The raise keyword forces an exception to occur. There are many different exceptions, but the one I forced to occur was SystemExit.	I used it to raise the SystemExit exception, which terminated the program, when the user was done.
SystemExit	It is an exception, that is used to call the sys.exit() function – which (as the name suggests) exits/terminates the program.	Using the raise function, I raise this exception which calls the sys.exit() function, which terminates the program.

Global	If a variable is declared inside a function, it is a local variable, and can only be used within that function. The 'global' keyword makes a variable a global variable, so that it can be used in other functions.	I use it to make the variable 'option' global, because I declared it in the main_menu() function, but I needed to call it in the feedback() function.
\n	Adds a new line. Usually used within the print function.	I used it to space out my code and make my code look neater.

Development Review

One of the requirements for the program was to ask the user's name and greet them by outputting their name. To solve this, I used the input() keyword, which allows the user to input a response to a question or an instruction given to them. To start off with, the code outputs a welcoming message to the user, and then asks them for their name. Here, I used the input keyword, and their input is then assigned to the variable called name, so every time the program has to address them by their name, we can use the variable to output whatever they inputted as their name. This part achieves the requirement of making it user-friendly. Immediately after, the user's name is used in a sentence, which shows them that the program recognizes them as a person, which is one of the things that set it apart from the others.

```
name = input("Name: ")#This will be where the user enters their name. This variable will be re-used in the code, and not only does it avoid repetition of asking for their name (
time.sleep(1)#See lines 2 and 5.
print("\nOkay " + name + ", we will give you a list of possible options. Remember, we are currently limited to only a few solutions, but hope to expand in the future. Please re
```

The screenshot demonstrates a clear use of this.

Another requirement was for the program to be able to ask them a series of questions, with different outcomes. To solve this, I created a variable called option, and then created if statements that depended on whatever was inputted for the variable option. For example, if the user entered the number 4, then the if statement for number 4 would be executed, and the outcome would be different depending on the number entered. The code outputs a list of options with the 'print' keyword, and asks the user to input a number, which is assigned to a variable called 'option' that is then made global with the 'global' keyword, to make it usable/callable in other parts of the code. This is the code, and the whole of it can be seen in the section below with the screenshots of my whole code.

```
def main_menu():#This is the main menu, where the user will be able to select their option. They will be taken back here if they choose to return to the main menu. These will f
    print("\n1.Your phone won't turn on \n2.Your phone won't charge \n3.Your phone's touch screen won't work \n4.Your phone does not play any sound. (This can include music or

    option = input("\nOption: ") #This will allow the user to input their option number, based on what they need to troubleshoot.
    global option #The variable option is a local variable. By using the global technique, it make the variable a global variable, meaning that it can be called in other functi

    if option == "1":#See line 27.
        print("\nYour phone not turning on could be for a number of reasons. First, make sure it is charged. Leave it pluged in for at least 5-10 minutes before trying to turn
        time.sleep(5)#See lines 7 and 9.
        feedback()#This calls the feedback() function, which is above in the code.

    elif option == "2":#See line 26.
        print("\nTry using another cable and power adapter that you know works on another phone, to see if the problem lies with your phone or your charger. If your phone charg
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "3":#See line 26.
        print("\nIf you find your touch screen unresponsive, try starting with turning your device off and on again. If the touch screen only unresponsive in certain places, tr
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "4":#See line 26.
        print("\nMake sure you turn up your volume. This can be done by pressing the volume keys usually located on the side of your device. If not, try going to settings and l
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "5":#See line 26.
        print("\nSim Card: Make sure you have a SIM card inserted first! Unless you are attempting to call by internet, in which case, see option number 7. \nCredit: Make sure
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "6":#See line 26.
        print("\nSim Card: Make sure you have a SIM card inserted first! Unless you are attempting to text by internet, in which case, see option number 7. \nCredit: Make sure
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "7":#See line 26.
        print("\nMobile data: If you're using your mobile data (3G/4G usually), ensure you have netowork coverage (Usually displayed in a top corner of your phone) and that you
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.

    elif option == "8":#See line 26.
        print("\nThe first thing to do is check your screen brightness. There are different ways to do this, depending on your phone. For example, on an iPhone, you swipe upwar
        time.sleep(5)#See lines 7 and 9.
        feedback()#See line 85.
```

And the output below.

```
1.Your phone won't turn on
2.Your phone won't charge
3.Your phone's touch screen won't work
4.Your phone does not play any sound. (This can include music or ringtones)
5.Your phone is unable to place calls.
6.Your phone is unable to send text messages
7.Your phone is unable to connect to the internet
8.Your phone's display is dim.
9.Your phone's microphone isn't working properly
10.Your problem is not listed above.

Option: 4

Make sure you turn up your volume. This can be done by pressing the volume keys usually located on the side of your device. If not, try going to settings and looking for the op
tion relating to sound.
```

However, if the user didn't enter a number between 1 and 10, it would ask them to do so and bring out the options list again. This is the code. The else statement ran if the input wasn't between 1 and 10.

```
else:#The ELSE statement is only run if all the IF statements aren't satisfied.
    print("\nPlease enter a number between 1 and 10.")#See line 58.
    main_menu()#See line 71.
```

And took it back to this part of the code.

```
def main_menu():#This is the main menu, where the user will be able to select their option. They will be taken back here if they choose to return to the main menu. These will f
    print("\n1.Your phone won't turn on \n2.Your phone won't charge \n3.Your phone's touch screen won't work \n4.Your phone does not play any sound. (This can include music or

    option = input("\nOption: ") #This will allow the user to input their option number, based on what they need to troubleshoot.
    global option #The variable option is a local variable. By using the global technique, it make the variable a global variable, meaning that it can be called in other functi
```

This made the code more efficient; it reminded the user to enter a valid number instead of simply crashing or terminating, so in a way, this met the requirement of having a smooth code with no errors, not mentioned explicitly, but to be expected by users of my program.

The program also had to use the answers given to try and diagnose the problem, so I had to use more if statements which involved things like asking the user if their problem was solved or if they needed more advice, ultimately telling them to refer to their manufacturer for further advice. To make sure the appropriate solution was given, the options were numbered, so they could enter a number and get the solution that was right for them. It also had to maintain a sense of user-friendliness, and it did so by often addressing the user by their name (or whatever was inputted) and by frequently asking questions to check if they were getting the right help they needed. I put the while loop that asked the user if their problem was solved in a function, so that each time the program needed to ask them that question, it simply had to call the function instead of repeating the code for the loop again.

```
def feedback():#This is where a function is defined, to avoid repetition in the code. This way, you simply have to call this function when needed, instead of rewriting it.
    feedback = 0#The variable must be declared before being used in the while loop, so here it gets declared, but with no value since it isn't needed yet.
    while feedback != "y" or feedback != "n":#The while loop here means that until the answer is y or n, it will keep repeating.

        feedback = input("\nDid this solve your problem? (y/n)")#This is where the user gets to input an answer. Since it only accepts the inputs y and n, it will keep asking tl
        feedback_lower = feedback.lower()#This converts the user's input into lowercase, so that it can match the case the if statements were written in.

        if feedback_lower == "y":#This statement will be run if the variable feedback_lower has a value of "y" (given by user input).
            print("We are glad to have helped you solve your issue," + name + "!")#Based on what the user entered for their name, we will be able to re-use their input and ment

        elif feedback == "n":#If the IF statement isn't valid, the program will move to the next statement, which in this case is an elif statement or, "else if".
            if option == "1":#The IF statement has a condition that must be met in order to run. In this case, the variable called 'option' has to have a value of a 1 (in strin
                print("\nCheck your phone for any damage. Severe damage may prevent it from working, (e.g dropping it from a high place or any kind of water damage) and if this
```

This is the beginning of the loop. You can see the complete code below in the 'code' section.

And the output, showing how the criteria "leading to a solution or advice to call the supplier."

```
Did this solve your problem? (y/n)n

This could mean a problem with your speakers. Go to your phone's manufacturer or repair shop for further assistance.
```

If their problem was solved, then they were thanked instead.

```
Did this solve your problem? (y/n)y
We are glad to have helped you solve your issue,Shaun!
```

The user then got asked if they wanted to return to the main menu, which meets the criteria for the effectiveness. I used the raise systemexit keywords to quit the program, which is means that the program would terminate properly instead of being left running.

```
Return to main menu?(y/n) n

Have a nice day!
```

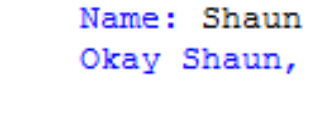
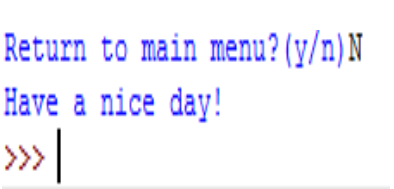
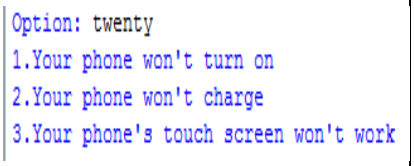
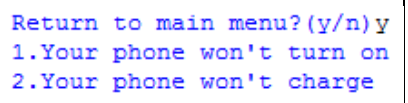
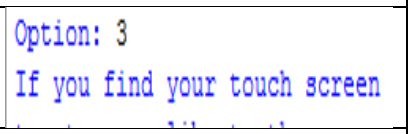
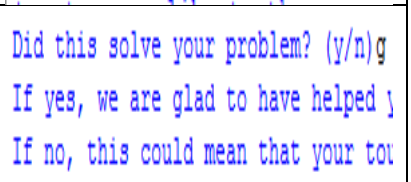
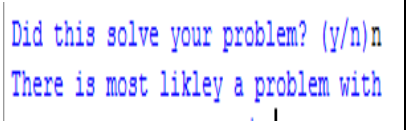
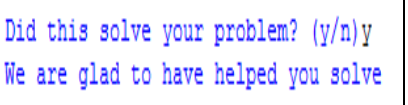
This would in turn help it stand out from the other programs and be the more favourable one to come to when you needed troubleshooting. To make it more organized, I used def functions to make my code more organized and less repetitive. For example, I put the while loop that asked the user if their problem was solved in a function, so that each time the program needed to ask them that question, it simply had to call the function instead of repeating the code for the loop again. I also used the global function, since I later decided that I would make the variable 'option' global, so that I could refer to the option in the feedback() function, since there were if statements that relied on the option number. For example, if their input was 1 for the option, and they replied "n" to the while loop that asked them if their problem got solved, then further information relating to option 1 would be outputted. This was the same for all the options, and it saved me having to create another variable to remember the user's input.

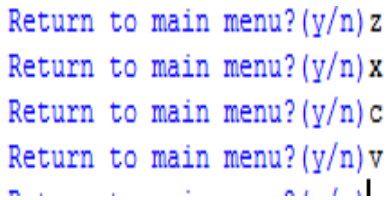
Code

```
#Centre number: 10854
#Candidate number: 7116
#Name: Shaun Sevume

import time #This will allow me to use the time.sleep function, which will make the program delay executing an insruction depending on how long I tell it to.
print("Welcome to the multicellular Troubeshooting program. If you have a problem with your phone, then you've come to the right place! Though we are only limited to a handful
time.sleep(2)#The program will wait two seconds before executing the next line as the time.sleep function delays the amount of time before the program executes the next line.
print("\nTo familiarize yourself with our program, we think it would be helpful if you enter your name.") #The \n makes a new line. It is used to make the text look neater.
time.sleep(1)#Delays the amount of time before the program executes the next line.
name = input("Name: ")#This will be where the user enters their name. This variable will be re-used in the code, and not only does it avoid repetition of asking for their name
time.sleep(1)#See lines 2 and 5.
print("\nOkay " + name + ", we will give you a list of possible options. Remember, we are currently limited to only a few solutions, but hope to expand in the future. Please re

time.sleep(1)#See lines 7 and 9.

def feedback():#This is where a function is defined, to avoid repetition in the code. This way, you simply have to call this function when needed, instead of rewriting it.
    feedback = 0#The variable must be declared before being used in the while loop, so here it gets declared, but with no value since it isn't needed yet.
    while feedback != "y" or feedback != "n":#The while loop here means that until the answer is y or n, it will keep repeating.

        feedback = input("\nDid this solve your problem? (y/n)")#This is where the user gets to input an answer. Since it only accepts the inputs y and n, it will keep asking t
        feedback_lower = feedback.lower()#This converts the user's input into lowercase, so that it can match the case the if statements were written in.

        if feedback_lower == "y":#This statement will be run if the variable feedback_lower has a value of "y" (given by user input).
            print("We are glad to have helped you solve your issue," + name + "!")#Based on what the user entered for their name, we will be able to re-use their input and ment

        elif feedback == "n":#If the IF statement isn't valid, the program will move to the next statement, which in this case is an elif statement or, "else if".
            if option == "1":#The IF statement has a condition that must be met in order to run. In this case, the variable called 'option' has to have a value of a 1 (in strin
                print("\nCheck your phone for any damage. Severe damage may prevent it from working, (e.g dropping it from a high place or any kind of water damage) and if this

            elif option == "2":#See line 26.
                print("\nThere is most likley a problem with your charging port. You will have to take your phone to the manufacturer to get them to sort it out.") #Further hel

            elif option == "3":#See line 26.
                print("\nThis could mean that your touch screen is broken, and you will have to get it replaced. If you do not wish to pay for a new touch screen, you can first

            elif option == "4":#See line 26.
                print("\nThis could mean a problem with your speakers. Go to your phone's manufacturer or repair shop for further assistance.")

            elif option == "5":#See line 26.
                 print("\nThis could mean a problem with your phone or network provider. could mean a problem with your phone or network provider. Contact your network provider

            elif option == "6":#See line 26.
                 print("\nThis could mean a problem with your phone or network provider. Contact your network provider about the issue after making sure your phone isn't the pr

            elif option == "7":#See line 26.
                print("\nMobile data: Contact your netowork provider about your issue. \nWifi: Contact your ISP about your issue.")

            elif option == "8":#See line 26.
                print("\nThis could mean possible damage to your screen, possibly the light filters. This damage is repairable, but you will have to go to your phone's manufact

            elif option == "9":#See line 24
                print("\nIf your phone has recently been dropped or has sustained water damage, this could cause problems with the microphone. However, it could simply be an ap

        return_toMenu()#Here, the return_toMenu() function is called.

def the_end():#See line 14
    time.sleep(1)#See lines 2 and 5
    print("\nHave a nice day!") #The print function is used to output text to the user.
    raise SystemExit #This terminates the program properly, so it isn't left running with nothing to display and nothing to be inputted.

def return_toMenu():#See line 14
    time.sleep(1)#See lines 2 and 5
    return1 = 0 #Before putting a variable in a while loop, this was necessary so that it was at least defined beforehand to prevent variable errors.

    while return1 != "y" or return1 != "n": #In this while loop, it will keep asking the user to return to the main menu until a valid input is entered.

        return1 = input("\nReturn to main menu?(y/n)")#See line 18
        return1_lower = return1.lower()

        if return1_lower == "y":#See line 25
            main_menu()#Here, the main_menu() function is called.

        elif return1_lower == "n":#See line 24
            the_end()#Here, the the_end() function is called.

def main_menu():#This is the main menu, where the user will be able to select their option. They will be taken back here if they choose to return to the main menu. These will f
    print("\n1.Your phone won't turn on \n2.Your phone won't charge \n3.Your phone's touch screen won't work \n4.Your phone does not play any sound. (This can include music or

    option = input("\nOption: ") #This will allow the user to input their option number, based on what they need to troubleshoot.
    global option #The variable option is a local variable. By using the global technique, it make the variable a global variable, meaning that it can be called in other functi

    if option == "1":#See line 25
        print("\nYour phone not turning on could be for a number of reasons. First, make sure it is charged. Leave it pluged in for at least 5-10 minutes before trying to turn
        time.sleep(5)#See lines 2 and 5
        feedback() #This calls the feedback() function, which is above in the code.

    elif option == "2":#See line 24
        print("\nTry using another cable and power adapter that you know works on another phone, to see if the problem lies with your phone or your charger. If your phone charg
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "3":#See line 24
        print("\nIf you find your touch screen unresponsive, try starting with turning your device off and on again. If the touch screen only unresponsive in certain places, tr
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82
```

```
    elif option == "4":#See line 24
        print("\nMake sure you turn up your volume. This can be done by pressing the volume keys usually located on the side of your device. If not, try going to settings and l
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82


    elif option == "5":#See line 24
        print("\nSim Card: Make sure you have a SIM card inserted first! Unless you are attempting to call by internet, in which case, see option number 7. \nCredit: Make sure
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "6":#See line 24
        print("\nSim Card: Make sure you have a SIM card inserted first! Unless you are attempting to text by internet, in which case, see option number 7. \nCredit: Make sure
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "7":#See line 24
        print("\nMobile data: If you're using your mobile data (3G/4G usually), ensure you have netowork coverage (Usually displayed in a top corner of your phone) and that you
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "8":#See line 24
        print("\nThe first thing to do is check your screen brightness. There are different ways to do this, depending on your phone. For example, on an iPhone, you swipe upwar
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "9":#See line 24
        print("\nMake sure you know where your microphone is, as the location varies with devices. If you can't find it, look in the manual that should've come with your phone
        time.sleep(5)#See lines 2 and 5
        feedback()#See line 82

    elif option == "10":#See line 24
        print("\nWe are very sorry to see that our program is unable to solve your problem," + name + ". In the future, we hope to improve it to solve an as wide range of proble
        return_toMenu()#See line 82

    else:#The ELSE statement is only run if all the IF statements aren't satisfied.
        print("\nPlease enter a number between 1 and 10.")#See line 56
        main_menu()#See line 68

main_menu()#The whole code is indented so that when it reads the line main_menu() it goes back to the top. Since up until now the whole code has been defined as the main menu, t
```

4. Testing

Test no.	Testing	Code to be tested	Input, if applicable	Expected Output	Actual
1	Does it address the user by their name based on their input?	name = input("Name: ")	Shaun	Okay, Shaun…	
2	Does it recognize inputs in both lower and upper case when asking to return to the main menu?	return1 = input("Return to main menu?(y/n)")	y,Y/n,N	Returns to the main menu if y or Y and ends if n or N	
3	Does it crash when something unexpected is inputted?	option = input("Option: ")	twenty (the variable option only has if statements from numbers 1-10	Error	 (etc…)
4	Is my code correctly indented?	if return1 == "y" or return1 == "Y": print(main_menu())	y	(main menu prints)	 (etc…)
5	Are variables correctly defined?	option = input("Option: ")	3	"If you find your touch screen…	
6	Is there an else statement in case the if statements are not met?	else: print("If yes…")	(anything that isn't y,Y/n,N)	If yes… If no…	
7	Does it give further advice if issue wasn't solved?	option2 = input("Did this solve your problem?")	n,N	There is most likely a problem with…	
8	Does it give an appropriate output based on the input?	option5 = input("Did this solve your problem?")	y,Y	We are glad to have helped you…	

9	Does it keep on asking for an input for a specific variable before a condition is met? (Iteration)	while return1 != "y" or return1 != "Y" or return1 != "n" or return1 != "N": return1 = input("Return to main menu?(y/n)")	(anything that isn't y,Y/n,N)	Return to main menu?(y/n)	

Testing review

My testing was very successful. I was able to run all of my tests without encountering a single error, due to me fixing all of them during my development. All my expected outcomes were produced, as shown in my screenshots, but I didn't have any new tests to record in my table.

5. Evaluation

My solution effectively solves the problem, able to produce 10 solutions for the most common phone problems. Evidence for this is the fact that when you enter a number between 1 and 10 when you are prompted to with the variable "option" if statements will be used to make sure that you only get a solution relevant to your input (e.g. if you entered 3 for a broken touch screen, it will give you advice for the screen only, not something else like the volume for example). It also satisfies all of the points in the success criteria, and screenshots of this can be seen in the testing table as it shows you the points one by one, and how they are satisfied in the code. I think my solution is fairly efficient, as it requires little user input and more reading. This makes the program easy to use and understand, since the input process isn't too complex. It always stays relevant to the user's problem and because it satisfies all of the success criteria, this makes it even more effective, as it ticks all the boxes in order to be a good program. My pseudocode and flowchart was based off my code, which I thought would be easier. Because of this, no changes were needed to be made to the two, since the code was already perfected and all possible errors had been fixed or prevented. Any changes made to my code to make it more efficient are listed in the development section, as they all helped the code to become better. I did some research and I found 3 troubleshooting programs. All three of them were webpage based, but rather than asking for user input, they required a lot of reading and clicking to find the specific problem they wanted. My code isn't like this however. The options are kept short and simple, and you only have to enter a number to bring up the solution, instead of navigating the site manually by clicking. Therefore, my program is more efficient than these, and I think that it would be favoured amongst the others, if they were to be chosen from. I have included the links to the sites in the references sections. My pseudocode and flowchart represent my code, and I didn't have to make any changes because I did agile programming instead of the waterfall lifecycle, as I found it easier to make the flowchart and pseudocode after the code had been made.

Limitations

During my development, at first my computer was unable to run python, so it was hard for me to make progress on my code, as I had to use an online website called repl.it, but it meant that I had to keep pasting my work from there to python and vice-versa. However, I managed to get a new computer, which could run python, making developing my code much more easier. My hardware was fine throughout, even on the old pc, as the reason python would not install was more of a software issue. My mouse worked fine, able to select and highlight with ease, and my keyboard was sufficient enough to use to type, though the new one that came with the new pc made it significantly easier.

Further development

In the future, to improve my code, I would make more solutions available, much more than 10. This would be relatively easy to do, as I would just have to add another elif statement and enter the solution from there, since the statements were quite repetitive. However, this is quite ineffective, so implementing txt files to keep track of solutions would be a better alternative. I would also try to add a function where you could enter your device make and model, so it could give advice specific to that model.

6.Conclusion

I was required to create a troubleshooting program for mobile phones, making it user friendly and easily understandable, targeted at people of a 12+ age. My program was supposed to be able to ask the user for their name, and using a range of questions, help them diagnose what's wrong with their phone. If the problem wasn't able to be solved, then the program would give you further advice was along the lines of "contact your supplier regarding (insert issue here)". It was also supposed to be user-friendly and easy to use. I decided to search for the top 10 most common phone solutions to use in my code, recalling what I had been taught in lesson to make my code. I also researched some new techniques, as mentioned in the development section. All of my research was done on the internet across various websites, which all helped me to develop my code into what it is now. My program will be able to help a user having problems with their phone, by being able to solve some of the most common problems. It also gives further advice, should it be unable to solve the user's issue, so that they weren't completely left in the dark as to what to do. This would improve their chances of getting their problem solved, which they would like a lot.

7.References

All of my solutions came from this one website http://www.unlox.co.uk/Repairs/mobile-phone-faults.asp , combined with my own knowledge of phones and my personal experience with them.

UNLOX.co.uk

Mobile Phone Unlocking Codes.....

Mobile Phone unlock codes for Alcatel BlackBerry Huawei Motorola Nokia Pantech Samsung and ZTE.

Free the network lock today.... From: £0.99

Home About Contact Testimonials SEARCH

Unlock By Codes

- Alcatel Codes
- Blackberry Codes
- Doro Codes
- Huawei Codes
- iPhone Codes
- Motorola Codes
- Nokia Codes
- Nokia BB5 Codes
- Pantech Codes
- Samsung Codes
- VeryKool Codes
- ZTE Codes

Unlock By Post

- Unlocking By Post From £12.50

Other Links

- Free iPhone Unlock
- Link To unlox
- Mobile Resources
- Payment Methods
- Recommend Our Site
- Secret Phone Codes
- Supported Unlock's
- Unlocking FAQ's
- Unlox News

Follow Us

Payment Methods

Secure Server

Mobile Phone Faults And Symptoms:

Detailed below are some of the most common mobile phone faults and the symptoms of these faults. Click on a fault for a more detailed explanation:

Battery / Charging Faults

Mobile phone battery or charging faults will usually occur with the following symptoms:

- The battery will not charge.
- The back plate will not go back on because battery is swollen.
- The battery only stays charged for a short while.

Microphone Faults

Mobile phone microphone faults will usually occur with the following symptoms:

- The caller cannot hear you.
- Your voice is distorted to the caller.
- The caller can hear you on some occasions but not all the time.

Speaker / Sound Faults

Mobile phone speaker faults will usually occur with the following symptoms:

- You cannot hear ringtones and/or music.
- Ringtones and/or music is distorted or difficult to understand.
- Ringtones and/or music is too quiet.

Earpiece / Hearing Faults

Mobile phone earpiece faults will usually occur with the following symptoms:

- You cannot hear the caller.
- The callers voice is distorted or difficult to understand.
- The callers voice is too quiet.

Charging Faults

Mobile phone charging faults will usually occur with the following symptoms:

- You are unable to charge the phone.
- The phone charges for a short period of time only.
- The charger will not fit correctly into the charging block.
- The phone shows a full charge but the charge lasts only a short time.

LCD (Liquid Crystal Display) Faults

Mobile phone LCD faults will usually occur with the following symptoms:

- The LCD is showing no image/text at all.
- The LCD is white.
- The image/text on the LCD is upside down or inverted.
- The image/text on the LCD appears intermittently.

KeyPad Faults

Mobile phone KeyPad faults will usually occur with the following symptoms:

- You are unable to type any numbers/letters.
- You can type some numbers/letters, but not all.
- Typing one number/letter results in a different number/letter being displayed.
- Numbers/letters appear on-screen without the KeyPad being pressed.
- Numbers/letters get stuck resulting in the number/letter being repeated several times on-screen.

Water Damage

Mobile phone water damage faults will usually occur with the following symptoms:

- If your mobile phone has been exposed to water, remove the battery and **DO NOT** switch it on as this could cause irreversable damage to the phone.
- A water damaged phone does not have any specific symptoms because it could cause any of the effects listed on this page.

Software or Firmware Faults

Mobile phone software/firmware faults will usually occur with the following symptoms:

- The phone switches itself off intermittently.
- The phone halts/freezes.
- Certain phone menus/features are no longer accessible.
- The screen flickers on and off.

These are just some of the most common mobile phone faults. If your phone has developed a fault not described above you will need to seek specialist advice!

Sitemap - Terms & Conditions - Privacy Policy

I also visited another website called stackoverflow.com (see development for screenshot).