Malware Analysis Management System Testing

Pablo Collado Soto
19134031

Introduction

For the first coursework in this module we were tasked with coding a simple record management system in assembly. After writing all the code we need to validate it in some way to be sure we did our job correctly. That's why we have carried out the following tests. We'll include a short description about each of them stating what they intend to test, the reason the test was successful (or not) and a screenshot of the output.

Tests

User record tests

Test 1: Initial conditions

Once the program is started we'll be greeted with a menu. The system should initially contain no users or computer records. Thus, actions such as deleting, searching for and listing the users should be aware of that fact. The initial user and computer counts should be correctly initialized as well.

In order to test it we only need to try and choose the different options and inspect their outcome. We'll include a screenshot for the user data and another for the computer data as we don't want to bloat the report with too many screenshots. Remember we implemented a function clearing the screen for our program.

We then see how the initial conditions in our program are behaving correctly.

```
root@4ff86c902894:/mw_repo/management_system#
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 2
There are no users currently on the system!
```

```
root@4ff86c902894:/mw_repo/management_system# ./main.ex
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 10
We currently have 0 registered computer(s) on the system!
```

Test 2: Validating the user data

As the specifications we were provided instructed us to take several rules into account regarding what comprised valid data we had to validate all the user input. These involve:

- The surname and last name need to be at most 64 characters long.
- The department must be one of HR, Finance, IT Support or Development.
- The user ID must begin with a lowercase p (‘p’) and have 7 digits afterwards.
- The e-mail address was NOT enforced to be read from the user so we automatically build it by concatenating the User ID and the provided email ending.

We are attaching a screenshot showing invalid input data and the associated error messages. We’ll test willingly bad inputs and then provide good ones to show the program’s progression:

We then see how the output is corrected when possible (the surname) and that new input is asked for when no recovery from the provided data is possible. We’re also attaching a screenshot showing how user IDs are forced to be unique. Right now user ‘p1111111’ is registered to the system so if we try to reuse that ID before deleting that entry we’ll stumble upon an error as registering the ID again would violate the uniqueness of the scheme. As searching and other functions depend on it we want to make sure duplicate IDs are NOT possible.

```
root@4ff86c902894:/mw_repo/management_system# ./main.ex
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 1
Please enter a surename -> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
The input string was too long and has been chopped down to 64 bytes!
Please enter a first name -> Pablo
Please enter a department -> foo
This department name is NOT supported... Choose one of:
       Development
       IT Support
       HR
       Finance
Please enter a department -> Finance
Please enter a User ID -> c1212121
Invalid initial letter!
Please enter a User ID -> p1
The ID is NOT 8 characters long... Input a new one!
Please enter a User ID -> p11111111
The ID is NOT 8 characters long... Input a new one!
Please enter a User ID -> p1111111
The email has been automatically generated based on the user ID!
```

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 1
Please enter a surename -> good_surename
Please enter a first name -> good_name
Please enter a department -> HR
Please enter a User ID -> p1111111
The ID has already been registered!
Please enter a User ID ->
```

Test 3: Printing the users

We just need to print the currently registered users to check whether data is being stored and displayed correctly. We can do so through option 4.

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 4
User number: 0
        Surename        -> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Name            -> Pablo
        Department      -> Finance
        User ID         -> p1111111
        E-mail          -> p1111111@helpdesk.co.uk
```

We find the data we had provided is correctly displayed.

Test 4: Searching for users

We'll put our search function to a test. In order to do so we'll try to look for an existent user and a nonexistent one and check the outputs. Remember user 'p1111111' is the only existent one.

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 3
Provide the User ID to look for -> p2222222
The requested user couldn't be found...
```

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 3
Provide the User ID to look for -> p1111111
        Surename        -> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Name            -> Pablo
        Department      -> Finance
        User ID         -> p1111111
        E-mail          -> p1111111@helpdesk.co.uk
```

We then see how the existent user is displayed whilst the non-existent one is correctly identified as well. We also show how we only accept valid user IDs, that is, they are also validated upon input.

Test 5: The user count

As our next test will involve deleting a user we have decided to check whether our user count is correct now. It should specify we have 1 registered user at the moment. We'll issue option 5 to check whether that's the case

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 5
We currently have 1 registered user(s) on the system!
```

We find the output is correct!

Test 6: Deleting a user

Given we are guaranteed that user IDs are unique we'll use them to index our user database and find the record to delete or, in case the provided record is nonexistent, print an error message informing of the situation. On success, the deleted user ID will be printed.

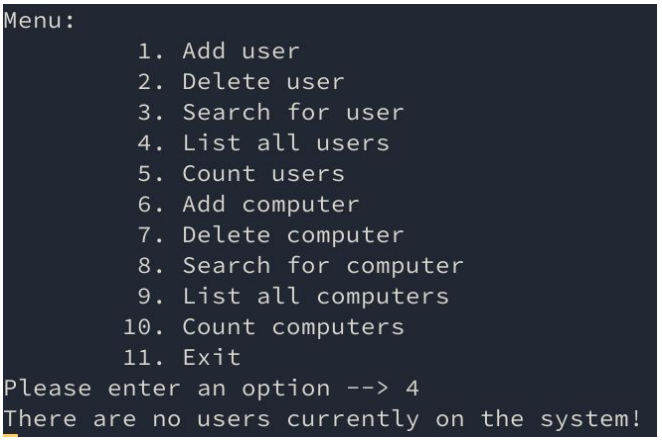
In order to test it we'll provide an invalid user IDs, a nonexistent one and a valid one (i.e. 'p1111111') to check whether functionality is as expected. We'll also include the list of users after the deletion.

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 2
User ID of the record to delete -> c121
Invalid initial letter!
User ID of the record to delete -> p1111111
Deleted -> p1111111
```

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 2
User ID of the record to delete -> p2222222
The requested user couldn't be found...
```

We then see how the User IDs need to be effectively validated, how correct ones pointing to real users will cause the deletion of these and how trying to delete valid but nonexistent IDs shows a suitable error message.

As we only had one user if we now try to print them we'll see the following output as there are no more registered users on the system.



That concludes the tests for the user records. We'll now move on into testing the computer records. As the operations we can perform on them are exactly the same as on the user data we'll find the most difference in how the data is validated for this new type of information. The functionality managing all these records has been coded in a "format agnostic" way in the sense that the pertinent subroutines can act on any type of records. Through a series of input parameters we can apply subroutines to the different record classes, hence increasing code reusability. Nonetheless, testing is a very sane practice and we'll test our computer records as thoroughly as we did with our data records to ensure there are no sneaky bugs lurking around.

Computer record tests

Test 7: Validating the computer data

In this case the data specifications are different but the essence of the problem remains. The following requirements are to be met:

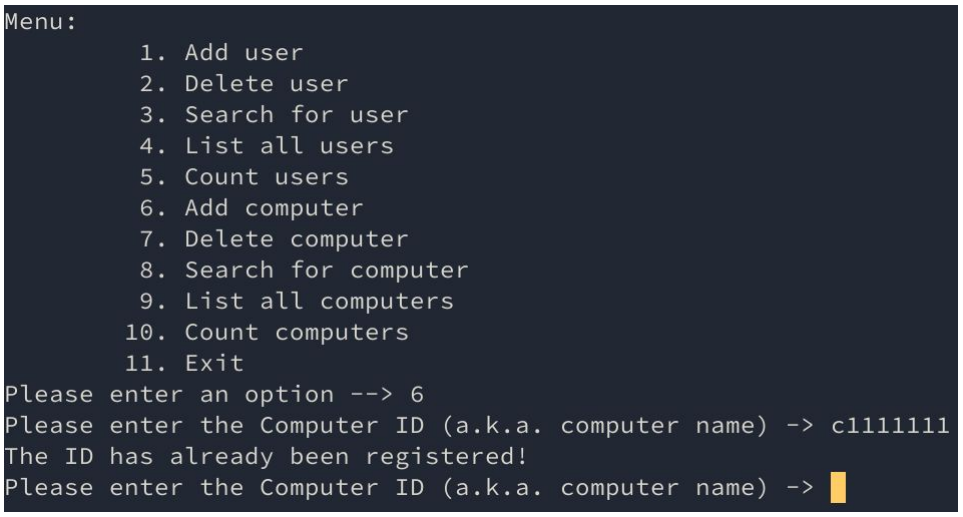
- The computer ID must begin with a lowercase c ('c') and have 7 digits afterwards.
- The IPv4 address should conform to the standard included in RFC 791. This roughly translates to having exactly 4 octets whose value is in the [0, 255] range and that are separated by a dot ('.'). Hence, "1.1.1.1" is a valid IP address whilst "1.256.1.1", "1..1.1.1" and "1.1.1.1.1" are NOT.

- The OS must be one of Linux, Windows, Mac OSX.
- The user ID must begin with a lowercase p (‘p’) and have 7 digits afterwards.
- The date must be valid. We need it to be provided in the dd/mm/yyyy format and we can also take leap years into account and allow February to have 29 days should the circumstances arise. We are also allowing dates that go as far as December 2020. We could only allow for past dates but that would mean modifying the source code whenever we wanted to update the “cutoff” date and given we are already validating elements such as leap years we deemed it not as interesting.

We are attaching a screenshot showing invalid input data and the associated error messages. We’ll test willingly bad inputs and then provide good ones to show the program’s progression:

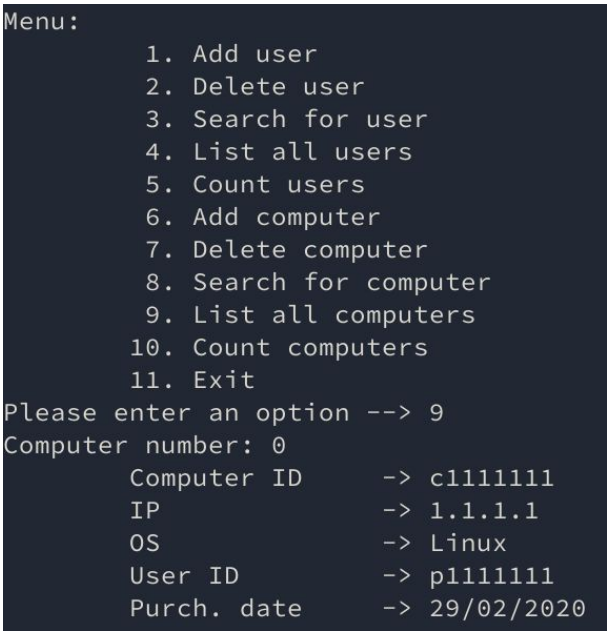
```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 6
Please enter the Computer ID (a.k.a. computer name) -> p1212121
Invalid initial letter!
Please enter the Computer ID (a.k.a. computer name) -> c1
The ID is NOT 8 characters long... Input a new one!
Please enter the Computer ID (a.k.a. computer name) -> c11111111
The ID is NOT 8 characters long... Input a new one!
Please enter the Computer ID (a.k.a. computer name) -> c1111111
Please enter the computer's IP -> 1.256.1.1
Invalid IP...
Please enter the computer's IP -> 1..1.1.1
Invalid IP...
Please enter the computer's IP -> 1.1.1.1.1
Invalid IP...
Please enter the computer's IP -> 1.1.1.1
Please enter the computer's OS -> foo
This OS name is NOT supported... Choose one of:
        Linux
        Windows
        Mac OSX
Please enter the computer's OS -> Linux
Please enter a this computer's user's User ID -> c
Invalid initial letter!
Please enter a this computer's user's User ID -> p1
The ID is NOT 8 characters long... Input a new one!
Please enter a this computer's user's User ID -> p11111111
The ID is NOT 8 characters long... Input a new one!
Please enter a this computer's user's User ID -> p1111111
Please enter the date of purchase (dd/mm/yyyy) -> 31/02/1999
The provided date is NOT valid!
Please enter the date of purchase (dd/mm/yyyy) -> 29/02/2019
The provided date is NOT valid!
Please enter the date of purchase (dd/mm/yyyy) -> 29/02/2020
```

We can see that validation is indeed working. Note 2020 has been a leap year and thus 2019 is NOT. If we were to try and register computer ID ‘c1111111’ again without deleting the associated user we would incur into an error due to the fact that they must be unique. The screenshot below demonstrates that behaviour:



Test 8: Printing the computer data

Just like with users we can indeed print the computer records. As we have just registered one before let’s see what the information has to offer.



We then see how the information we entered is correctly stored and shown where appropriate.

Test 9: Searching for computers

Given computer IDs are guaranteed to be unique given how we add new records we can use it as a key to univocally identify all the registered equipment. We'll try to look for the record we have just added showing how the computer ID will be validated and how, if we provide a nonexistent computer ID we'll get a message informing us of the situation.

In order to do that we'll try a badly formatted ID, and then that of the record we have just introduced. We'll then provide a valid but nonexistent ID and take a look at the result.

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 8
Provide the Computer ID to look for -> p
Invalid initial letter!
Provide the Computer ID to look for -> c1
The ID is NOT 8 characters long... Input a new one!
Provide the Computer ID to look for -> c11111111
The ID is NOT 8 characters long... Input a new one!
Provide the Computer ID to look for -> c1111111
       Computer ID     -> c1111111
       IP              -> 1.1.1.1
       OS              -> Linux
       User ID         -> p1111111
       Purch. date     -> 29/02/2020
```

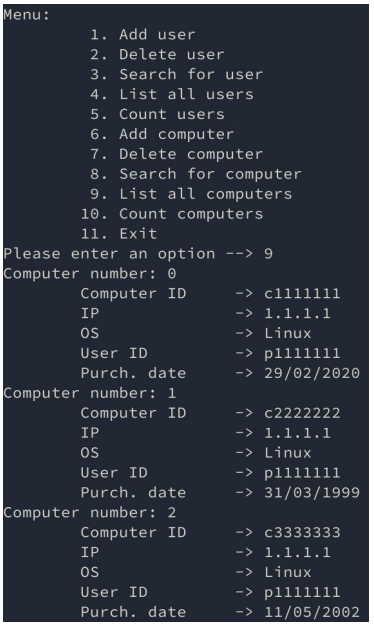
If we now provide a valid nonexistent ID we arrive to:

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 8
Provide the Computer ID to look for -> c2222222
The requested computer couldn't be found...
```

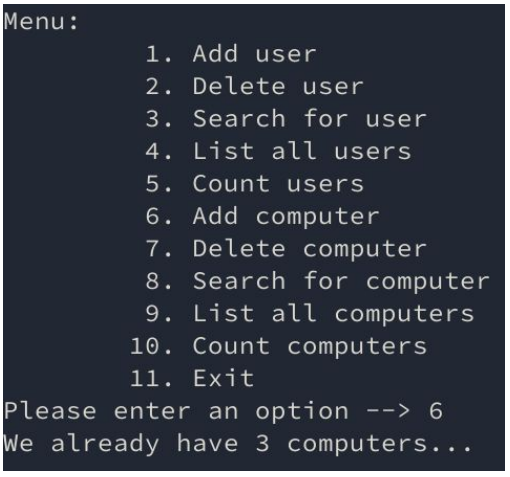
Test 10: Testing the limits of our record lists

The requirements for the coursework stated that we must be able to accomodate 100 user records and 500 computer records. As these values are too large to be tested experimentally we have manually set the computer list size limit to 3 so that we can see whether our code works as expected. We'll show the output of the printing option as well as that of the one letting us add computers once all 3 records have been filled.

To reach the limit we just need to add 2 computers on top of the one we already had.



Now if we try to add a new user:



We then see how our limiting code works. We only need to change the 'MAX_COMPS' constant within the 'comp_stuff.asm' file from '3' back to '500' to adhere to the requirements.

Test 11: Counting the computers

Given we have reached the computer limit we should check that the current number of registered records is actually 3. In order to do so we only need to invoke option '10' in the menu and check the output.

```
Menu:
        1. Add user
        2. Delete user
        3. Search for user
        4. List all users
        5. Count users
        6. Add computer
        7. Delete computer
        8. Search for computer
        9. List all computers
       10. Count computers
       11. Exit
Please enter an option --> 10
We currently have 3 registered computer(s) on the system!
```

We find that indeed we have taken into account all 3 computers that we have registered.

Test 12: Deleting computers

In order to delete computers we'll unequivocally identify them by their computer ID which will be validated. We'll then provide bogus computer IDs to check that validation is working appropriately and then provide a computer ID that's not been assigned to any machines. After that we'll provide a valid computer ID and erase a record. We'll then try to add a new record, which should be allowed given we have deleted one. What's more, the new record should occupy the place of the deleted one as our code is capable of reusing invalidated/deleted records. We'll then delete a second computer and reuse its ID to check it's effectively being freed once it's deleted. As these are a lot of tests we'll include several incremental screenshots showing the developments and we'll also label them. Please remember our current state is the one shown in test 10.

Checking for computer ID validation and trying to delete a non-existent computer:

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 7
Computer ID of the record to delete -> p
Invalid initial letter!
Computer ID of the record to delete -> c1
The ID is NOT 8 characters long... Input a new one!
Computer ID of the record to delete -> c111111111111
The ID is NOT 8 characters long... Input a new one!
Computer ID of the record to delete -> c8888888
The requested computer couldn't be found...
```

Deleting an existent computer record:

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 7
Computer ID of the record to delete -> c2222222
Deleted -> c2222222
```

Listing the users after the deletion:

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 9
Computer number: 0
        Computer ID     -> c1111111
        IP              -> 1.1.1.1
        OS              -> Linux
        User ID         -> p1111111
        Purch. date     -> 29/02/2020
Computer number: 2
        Computer ID     -> c3333333
        IP              -> 1.1.1.1
        OS              -> Linux
        User ID         -> p1111111
        Purch. date     -> 11/05/2002
```

Adding a new user. Note it'll occupy 'c2222222's place as it's the record we have deleted. That is, it'll be placed between 'c1111111' and 'c3333333'. We are including the record addition and the listing.

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 6
Please enter the Computer ID (a.k.a. computer name) -> c4444444
Please enter the computer's IP -> 1.1.1.1
Please enter the computer's OS -> Linux
Please enter a this computer's user's User ID -> p1111111
Please enter the date of purchase (dd/mm/yyyy) -> 31/03/1999
```

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 9
Computer number: 0
        Computer ID     -> c1111111
        IP              -> 1.1.1.1
        OS              -> Linux
        User ID         -> p1111111
        Purch. date     -> 29/02/2020
Computer number: 1
        Computer ID     -> c4444444
        IP              -> 1.1.1.1
        OS              -> Linux
        User ID         -> p1111111
        Purch. date     -> 31/03/1999
Computer number: 2
        Computer ID     -> c3333333
        IP              -> 1.1.1.1
        OS              -> Linux
        User ID         -> p1111111
        Purch. date     -> 11/05/2002
```

Now, if we delete 'c4444444' and 'c333333' and then try to add a new record using 'c3333333' we'll see there is no problem as the 'c333333' computer ID is freed for another computer to take as soon as it's deleted.

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 7
Computer ID of the record to delete -> c4444444
Deleted -> c4444444
```

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 7
Computer ID of the record to delete -> c3333333
Deleted -> c3333333
```

```
Menu:
         1. Add user
         2. Delete user
         3. Search for user
         4. List all users
         5. Count users
         6. Add computer
         7. Delete computer
         8. Search for computer
         9. List all computers
        10. Count computers
        11. Exit
Please enter an option --> 6
Please enter the Computer ID (a.k.a. computer name) -> c3333333
Please enter the computer's IP -> 1.1.1.1
Please enter the computer's OS -> Linux
Please enter a this computer's user's User ID -> p1111111
Please enter the date of purchase (dd/mm/yyyy) -> 31/03/1999
```

With that we have shown how the computer records are correctly deleted and the associated memory areas ready for being reused making the code efficient with regards to memory usage.

Test 13: Exiting the program

Just to be sure our program exits nicely we have invoked option 11 and we get a nice goodbye message before being returned

to the prompt.

With that, the program's working has been entirely tested.

Conclusion

We hope that these tests were thorough enough to account for all our program's functionality. If there is anything we missed please do let us know.