A close-up of a logo

Description automatically generated

Foundation Certificate of Higher Education

Module Code & Name: - **DOC333**: Introduction to Programming principles

Module Leader: - Mr. Sudarshan Welihinda

Assessment Type: - Individual Coursework

Submission Date: - 8.12.2023

Student:

|  |  |
| --- | --- |
| Student ID | Name of the Student |
| 20231297 | Sakindu Dehipitiya |

# **Acknowledgements**

I would like to extend my heartfelt thanks to our dedicated Module leader**, Mr. Sudarshan Welihinda** and our tutorial lecturer, **Ms. Tharushi Sandamali** for their guidance, support, and encouragement throughout this individual coursework. I was able to greatly benefit from the advice given by them regarding the preparation of the report and the python program.

I would like to express my profound appreciation to my fellow **batch mates and my family** for their consistent helpful hand and encouragement. I would like to conclude my thanks by appreciating everyone who supported me throughout this coursework.

Table of Contents

[Acknowledgements i](#_Toc152754503)

[1. Introduction 1](#_Toc152754504)

[1.1. The Problem 1](#_Toc152754505)

[1.2. Assumptions 2](#_Toc152754506)

[2. The Solution for the problem 3](#_Toc152754507)

[2.1. The Data Structure 3](#_Toc152754508)

[2.2. The Algorithm 4](#_Toc152754509)

[**2.2.1. How the process of adding a new project works in the solution** 11](#_Toc152754514)

[**2.2.2. How the process of removing a project from the existing projects works in the solution** 12](#_Toc152754515)

[**2.2.3. How the process of adding workers to the available workers group works in the solution** 13](#_Toc152754516)

[**2.2.4. How the process of updating an ongoing project works in the solution** 13](#_Toc152754517)

[**2.2.5. How the process of displaying statistics of projects works in the solution** 15](#_Toc152754518)

[**2.2.6. How termination of the interface is implemented** 16](#_Toc152754519)

[**2.2.7. How the process of checking whether a code entered by the user is unique works** 16](#_Toc152754520)

[**2.2.8. How the process of finding a project with a code entered by the user works** 17](#_Toc152754521)

[3. Test Cases 18](#_Toc152754522)

[3.1. Test case for the Main Menu 18](#_Toc152754523)

[3.2. Test cases for the “Add a new project” function 18](#_Toc152754524)

[**3.2.1. Test Case 1** 18](#_Toc152754525)

[**3.2.2. Test Case 2** 19](#_Toc152754526)

[3.3. Test case for the “Removing a completed project from existing projects” function 21](#_Toc152754527)

[3.4. Test cases for the “Add new workers to the available workers group” function 22](#_Toc152754528)

[3.5. Test cases for the “Updating an ongoing project” function 23](#_Toc152754529)

[3.6. Testing of the “Projects statistics” function 25](#_Toc152754530)

[Conclusion 26](#_Toc152754531)

# **List of Figures**

[Figure 1 The main data structure 3](#_Toc152755326)

[Figure 2 The data structure of an ongoing project or an on-hold project 3](#_Toc152755327)

[Figure 3 The data structure of a completed project 3](#_Toc152755328)

[Figure 4 A figure illustrating the "Add a new project" function 11](#_Toc152755329)

[Figure 5 A figure illustrating the "Add a new project" function 12](#_Toc152755330)

[Figure 6 A figure that demonstrates the "Removing a completed project from existing projects" function 12](#_Toc152755331)

[Figure 7 A figure illustrating the "Add new workers to the available workers group" function 13](#_Toc152755332)

[Figure 8 A figure demonstrating the "Add new workers to the available workers group" function 13](#_Toc152755333)

[Figure 9 When there are not enough available workers for the new number of workers required 14](#_Toc152755334)

[Figure 10 When there are enough number of available workers for the new number of workers 14](#_Toc152755335)

[Figure 11 When the new number of workers is less than the previous number of workers required 15](#_Toc152755336)

[Figure 12 Project Statistics function 15](#_Toc152755337)

[Figure 13 Terminating the interface 16](#_Toc152755338)

[Figure 14 The procedure of finding whether a code entered by the user is unique or not 16](#_Toc152755339)

[Figure 15 Finding a project in a specific list using a code 17](#_Toc152755340)

[Figure 16 Screenshot for the test case 1 of the 1st function 19](#_Toc152755341)

[Figure 17 Screenshot for the test case 2 of the 1st function 20](#_Toc152755342)

[Figure 18 Screenshot for the test case of the 2nd function 21](#_Toc152755343)

[Figure 19 Screenshot for the test case of the 3rd function 23](#_Toc152755344)

[Figure 20 Screenshot for the test case of the 4th function 24](#_Toc152755345)

[Figure 21 Screenshot of the "Project statistics" function 25](#_Toc152755346)

# **1. Introduction**

## **1.1. The Problem**

The problem for which a solution should be provided is the maintenance of the details of the projects a construction company named, “XYZ” undertake. As a solution, an information system was required to be made using the Python programming language. The company requires the following functions from it.

* **User Interface –** It was required to make an interface which at first, displays a main menu and an input field for the user to enter the user’s choice. Then, display helpful texts, inputs and instructions that are specific to the user’s choice.
* **Adding a new project –** There is one factor that can affect the status of the project that is going to be added. It is the current number of available workers in the company. If that number is less than the required number of workers, the status of the newly added project should be set to “on hold”. If that number is greater, the status should be set to “ongoing” and the required number of workers of the project should be subtracted from the number of available workers.
* **Removing a completed project from the existing projects –** An option should be able in the main menu to remove a completed project from the existing projects and record it as a completed project whenever it gets completed. When a project gets marked as completed, the number of workers that was utilized in that project will be released and available.
* **Adding new workers to the available workers group –** The company should be able to add new workers to the current number of available workers.
* **Updating details of an ongoing project –** As every project is saved with a code, the company should be able to find an ongoing project by the code and update the details.
* **Getting statistics of the projects –** The numbers of ongoing projects, on hold projects, completed projects and the number of available workers to assign to a project should be displayed.

## **1.2. Assumptions**

The assumptions made about the problem before the creation of the solution are mentioned as follows.

* **Uniqueness of the project code –** It was assumed that the code of every project should be unique throughout all the other projects (on hold, ongoing and completed). It has supported to effectively mark a project as completed and find and update a project with new details.
* **Unnecessariness of the “Project status” input –** It was assumed that project status handling should be automated, and the status of a project should not be directly decided by the user. It is because if the “Project status” field is provided to the user to input data in the “adding a new project” function and the “updating an ongoing project” function, that status can clash with the number of available workers. For instance, if the user enters a new number in the “updating an ongoing project” function which the current number of available workers is not sufficient for and the status as “ongoing”, some unnecessary confirmation and validation steps should be implemented. So, the “Project status” field has been omitted. Thus, the status handling has been automated.
* **Moving an on-hold project to ongoing projects after adding workers to the available workers group –** An assumption was made that the solution should be able to find the first eligible project in the on-hold project list and set its status as ongoing when the user adds workers to the number of available workers. For a project to be eligible, its required number of workers should be less than or equal to the new number of available workers. If an eligible project is not found, it just needs to add the user input number to the number of available workers.

# **2. The Solution for the problem**

A working solution has been implemented based on the given instructions and the assumptions stated above using the Python programming language. The data structure used to store the data about the projects, how that data is manipulated in the functions, the algorithm used in the solution and test cases that show the working state of the solution have been explained as follows.

## **2.1. The Data Structure**

A variable named ***companyData*** is initialized at the beginning of the script. The purpose of that variable is to store the number of available workers and all the project data that is essential for the solution. In the *companyData* variable, there are four items. The first item is a list which is used to store on hold projects. The second item is a list which is used to store ongoing projects. The third item is also a list which is used to hold data about completed projects. The fourth item is an integer which represents the number of workers available to assign to a project.

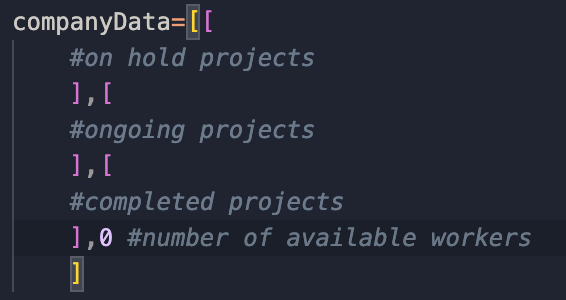


Figure 1 The main data structure



Figure 2 The data structure of an ongoing project or an on-hold project

*Figure 2* represents the data structure of a project that has the status as ongoing or on-hold. It is a which consists of the following five items respectively.

1. ***pCode –*** The code of the project
2. ***clientName –*** The name of the client of the project
3. ***startDate –*** The starting date of the project
4. ***expectedEnd –*** The expected ending date of the project
5. ***numOfWorkers –*** The number of workers required by the project



Figure 3 The data structure of a completed project

*Figure 3* represents the data structure of a completed project. It also consists of all the fields mentioned in the *Figure 2,* but it entails another field named ***actualEndDate***. It is used to store the date in which the project actually ended.

### **2.2.1. How the process of adding a new project works in the solution**

A white background with blue text

Description automatically generated

Figure 4 A figure illustrating the "Add a new project" function

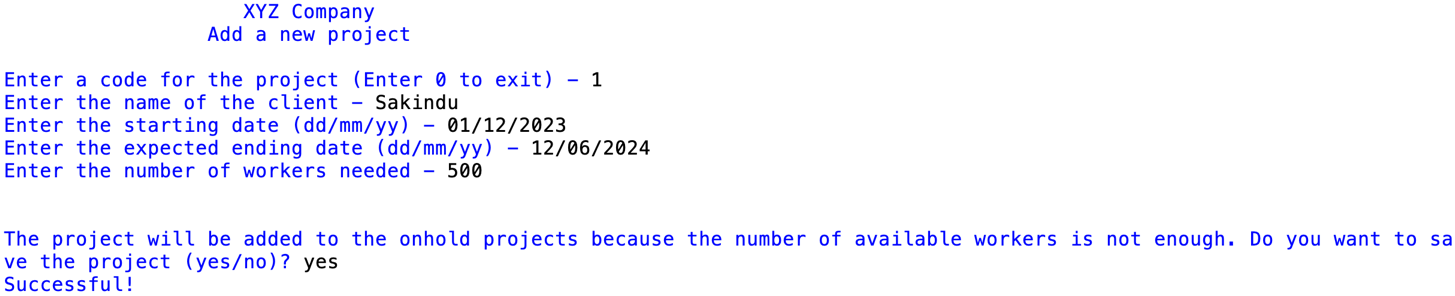


Figure 5 A figure illustrating the "Add a new project" function

In the process of adding a new project, the program checks whether the code entered by the user for the new project is unique throughout all the project lists (on hold, ongoing and on hold). If it is unique only, the process will continue. Then the program asks for the other relevant project data. If the number of workers required by the new project is less than or equal to the number of available workers, it asks for confirmation, adds the new project to the ongoing project list and subtracts the number of workers required by that project from the number of available workers (*Figure 4)*. If the number of workers required by the project is greater than the number of available workers, it asks for confirmation and adds the new project to the on-hold project list (*Figure 5)*.

### **2.2.2. How the process of removing a project from the existing projects works in the solution**

A white background with blue text

Description automatically generated

Figure 6 A figure that demonstrates the "Removing a completed project from existing projects" function

The program removes completed projects from ongoing projects by asking the user to enter the project code. It checks if any project exists with the user's code in the ongoing project list. If found, it displays the details of that project, adds the project to the completed list, and deletes the completed project data in the ongoing project list. The program also increments the number of available workers by the required number of workers of the completed project.

### **2.2.3. How the process of adding workers to the available workers group works in the solution**

A close-up of a sign

Description automatically generated

Figure 7 A figure illustrating the "Add new workers to the available workers group" function

In the process of adding new workers to the available workers group, there can be two main situations. If there is an on-hold project that is eligible to be moved to the ongoing project list because of requiring a number of workers less than or equal to the addition of the current number of available workers and the user input number, the program finds it and notifies the user that an eligible on-hold project is to be moved to the ongoing project list. If the user confirms, that on-hold project is removed from the on-hold project list and added to the ongoing project list. The number of available workers is incremented by the user input number and then the number of workers required by the onhold project that was moved to the ongoing project list in the current operation is subtracted from it. This operation is demonstrated in the *Figure 7.*

A group of blue text

Description automatically generated

Figure 8 A figure demonstrating the "Add new workers to the available workers group" function

The other situation is when an eligible project that can be moved to the ongoing project list is not found. If it occurs, the program will add the user input number to the number of available workers (*Figure 8*).

### **2.2.4. How the process of updating an ongoing project works in the solution**

At the beginning of the operation, the program asks for the code of the project the user wants to update. Then the program finds for a project with the entered code in the ongoing project list. If a project is found, the details of the project are displayed and asks for the new details. After getting the new code for the project, the program checks whether the code is unique and if it is unique or is the same as the previous code, the process will continue. If it continues, it will ask for the other details and the new number of workers required by the project. In this process, there can be two main situations.

A screenshot of a computer program

Description automatically generated

Figure 9 When there are not enough available workers for the new number of workers required

A screenshot of a computer program

Description automatically generated

Figure 10 When there are enough number of available workers for the new number of workers

The first situation is when the user enters a number greater than the previous number of workers required by the project, to the new number of workers. If the number of available workers is greater than or equal to the difference between the new number of workers and the previous number of workers of the project, the project will be updated and stored in the ongoing project list (as shown in *Figure 10).* Otherwise, that project is updated and stored as an on-hold project because there is not enough available workers *(Figure 9).*

A screenshot of a computer program

Description automatically generated

Figure 11 When the new number of workers is less than the previous number of workers required

The second situation is when the new number of workers is less than the previous number of workers required by the project. The program will increment the number of available workers by the difference between the previous number of workers and the new number of workers required by the project. The program will update and save the project in the ongoing project list (*Figure 11*).

If the new number of workers and the previous number of workers required by the project is the same, the program will just update and save the project in the ongoing project list.

### **2.2.5. How the process of displaying statistics of projects works in the solution**

A screenshot of a computer

Description automatically generated

Figure 12 Project Statistics function

As shown in the *Figure 12*, the program separately displays the numbers of the project lists. It finds the lengths of the three project lists individually and gets the number of available workers in the *companyData* list to display them.

### **2.2.6. How termination of the interface is implemented**

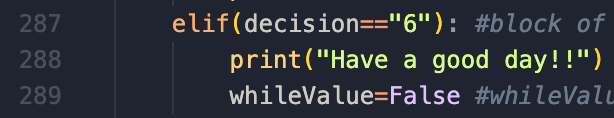


Figure 13 Terminating the interface

A variable named *whileValue* is declared and assigned the value *True* before the main while loop in the program. If the value of *whileValue* is *True* only, the loop will run. At the beginning of the while loop, it asks for the decision of the user in the main menu and after that, separate code blocks are written for the functions. If the user enters “6” as the decision, the code in the *Figure 13* will run which will assign the value *False* to the *whileValue* variable. As a result, the program will stop running.

### **2.2.7. How the process of checking whether a code entered by the user is unique works**

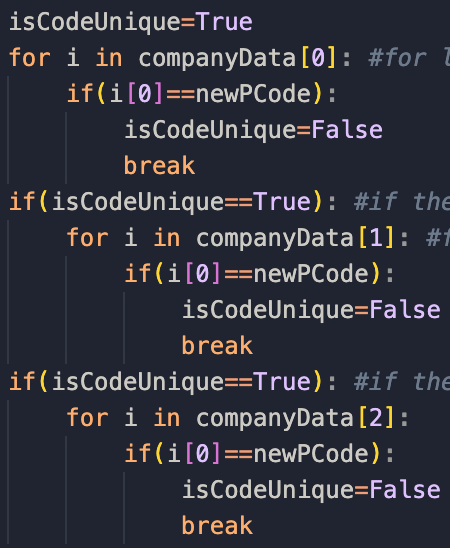


Figure 14 The procedure of finding whether a code entered by the user is unique or not

The code block shown in the *Figure 14* has been used in two functions in the interface. It is used to check if a given project code is unique or not. First, a variable named *isCodeUnique* is declared and assigned the value *True.* Then a For loop is run, going through every project in the on-hold project list. In that, if the code of the project in the current running is equal to the given code, *isCodeUnique* is set to *False* and the loop is terminated*.* If *isCodeUnique* is still set to *True* after the running of the previous for loop, another identical for loop is run through the ongoing project list. Another IF statement evaluates whether *isCodeUnique* is *True* to ensure that the given code is also unique in the completed project list. After the execution of the whole code block, if the code is unique, *isCodeUnique* is set to *True*. If it is not unique, *isCodeUnique* will be set to *False.*

### **2.2.8. How the process of finding a project with a code entered by the user works**

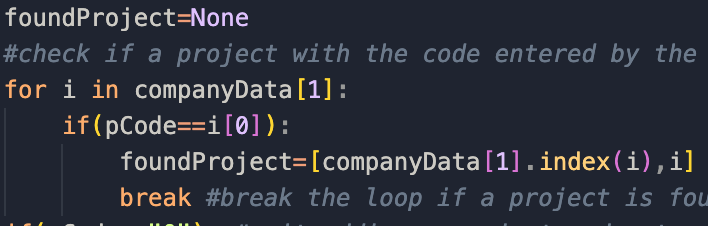


Figure 15 Finding a project in a specific list using a code

The code block shown in the *Figure 15* has been used in the second and fourth function of the solution. First, a variable named *foundProject* is declared and assigned the value *None* which is used to represent emptiness of a variable in Python. Then, a for loop is run on the project list which the project should be searched in. If the provided code is equal to the code of the project returned in the current running of that for loop, *foundProject* is assigned a list in which the first item is the index of the project found in the relevant list and the second item is the project found as a list. When a project is found, that for loop is terminated at that moment itself for more efficiency.

# **3. Test Cases**

## **3.1. Test case for the Main Menu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Outcome** | **Actual Outcome** | **Result** |
| 1 | Start “Adding a new project” process | Start “Adding a new project” process | Pass |
| 2 | Start “Removing a completed project” process | Start “Removing a completed project” process | Pass |
| 3 | Start “Adding new workers to the available workers group” process | Start “Adding new workers to the available workers group” process | Pass |
| 4 | Start “Updating details of an ongoing project” process | Start “Updating details of an ongoing project” process | Pass |
| 5 | Start “Displaying statistics of projects” process | Start “Displaying statistics of projects” process | Pass |
| 6 | Exit the interface | Exit the interface | Pass |
| -6 / abc | Displays “Invalid input” and returns to the main menu | Displays “Invalid input” and returns to the main menu | Pass |

## **3.2. Test cases for the “Add a new project” function**

### **3.2.1. Test Case 1**

Assume that the following processes were followed before making the test case 1 for the “Adding a new project” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Data inserted** | **Expected Outcome** | **Actual Outcome** | **Result** |
| Project code | “1” | Continues the process | Continues the process | Pass |
| Number of workers | 50 | Accepts the project and saves it to the ongoing project list | Accepts the project and saves it to the ongoing project list | Pass |
| Number of workers | -1 | Tells the user that number of workers cannot take a negative value | Tells the user that number of workers cannot take a negative value | Pass |

A computer code with blue text

Description automatically generated with medium confidence

Figure 16 Screenshot for the test case 1 of the 1st function

### **3.2.2. Test Case 2**

Assume that the following processes were followed before making the test case 2 for the “Adding a new project” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Data inserted** | **Expected Outcome** | **Actual Outcome** | **Result** |
| Project code | “0” | Returns the user to the main menu | Returns the user to the main menu | Pass |
| Project code | “1” | Notifies the user that a project with the code entered by the user already exists | Notifies the user that a project with the code entered by the user already exists | Pass |
| Project code | “2” | Accepts the code and continues the process | Accepts the code continues the process | Pass |
| Number of workers | 1500 | Accepts the project and saves it to the on-hold project list | Accepts the project and saves it to the on-hold project list | Pass |
| Number of workers | -1 | Tells the user that number of workers cannot take a negative value | Tells the user that number of workers cannot take a negative value | Pass |

A blue text on a white background

Description automatically generated

Figure 17 Screenshot for the test case 2 of the 1st function

## **3.3. Test case for the “Removing a completed project from existing projects” function**

Assume that the following processes were followed before making the test case for the “Removing a completed project” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Data inserted** | **Expected Outcome** | **Actual Outcome** | **Result** |
| Project code | “2” | Notifies the user that a project could not be found and return to main menu | Notifies the user that a project could not be found and return to main menu | Pass |
| Project code | “1” | First, asks for the actual end date of the project. Then, asks for confirmation and move the project to completed projects | First, asks for the actual end date of the project. Then, asks for confirmation and move the project to completed projects | Pass |

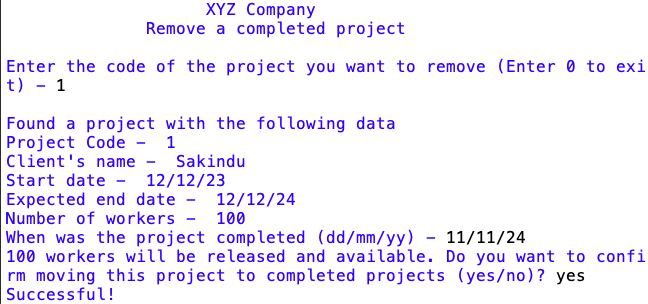


Figure 18 Screenshot for the test case of the 2nd function

## **3.4. Test cases for the “Add new workers to the available workers group” function**

Assume that the following processes were followed before making the test case for the “Removing a completed project” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Data inserted** | **Expected Outcome** | **Actual Outcome** | **Result** |
| Increment value of number of available workers | 0 | Checks for any on-hold project that can be moved to ongoing projects. If a project is found, move it after asking for confirmation. | Checks for any on-hold project that can be moved to ongoing projects. If a project is found, move it after asking for confirmation. | Pass |
| Increment value of number of available workers | -1 | Notifies that the number of workers cannot be negative | Notifies that the number of workers cannot be negative | Pass |
| Increment value of number of available workers | 1200 | Shows that the on-hold project with the code-1 can be moved to ongoing. It asks for confirmation, increments the number of available workers, and moves that project to the ongoing project list. | Shows that the on-hold project with the code-1 can be moved to ongoing. It asks for confirmation, increments the number of available workers, and moves that project to the ongoing project list. | Pass |
| Increment value of number of available workers | 500 | As there is no on-hold project that can be moved to ongoing, it just increments the number of available workers by the given value | As there is no on-hold project that can be moved to ongoing, it just increments the number of available workers by the given value | Pass |

A close-up of blue text

Description automatically generated

Figure 19 Screenshot for the test case of the 3rd function

## **3.5. Test cases for the “Updating an ongoing project” function**

Assume that the following processes were followed before making the test case for the “Updating details of an ongoing project” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Data inserted** | **Expected Outcome** | **Actual Outcome** | **Result** |
| Code of the project the user wants to update | “0” | Returns to the main menu | Returns to the main menu | Pass |
| Code of the project the user wants to update | “232” | Notifies that a project could not be found with the code given and returns to the main menu | Notifies that a project could not be found with the code given and returns to the main menu | Pass |
| Code of the project the user wants to update | “1” | Notifies that a project was found and continues the process. | Notifies that a project was found and continues the process. | Pass |
| New code for the project | “1” | Accepts the code and continues the process | Accepts the code and continues the process | Pass |
| New code for the project | “3” | Accepts the code and continues the process | Accepts the code and continues the process | Pass |
| New code for the project | “2” | Rejects the code as it is not unique, notifies the user about it and returns to the main menu | Rejects the code as it is not unique, notifies the user about it and returns to the main menu | Pass |
| New code for the project | “0” | Returns to the main menu | Returns to the main menu | Pass |
| New number of workers | 1100 | Notifies that the project will be moved to on-hold projects and moves it. | Notifies that the project will be moved to on-hold projects and moves it. | Pass |
| New number of workers | 750 | Makes the project remain in the ongoing project list | Makes the project remain in the ongoing project list | Pass |
| New number of workers | 500 | Makes the project remain in the ongoing project list | Makes the project remain in the ongoing project list | Pass |
| New number of workers | 250 | Makes the project remain in the ongoing project list and releases 250 workers to the available workers group | Makes the project remain in the ongoing project list and releases 250 workers to the available workers group | Pass |
| New number of workers | 0 / -1 | Notifies that the new number of workers cannot take zero or a negative value and returns to the main menu | Notifies that the new number of workers cannot take zero or a negative value and returns to the main menu | Pass |

A screenshot of a computer program

Description automatically generated

Figure 20 Screenshot for the test case of the 4th function

## **3.6. Testing of the “Projects statistics” function**

Assume that the following processes were followed before the testing of the “Project statistics” function. The data that were given as input in a particular process have been ordered according to the order of the inputs in that process.

A blue text on a white background

Description automatically generated

Figure 21 Screenshot of the "Project statistics" function

# **Conclusion**

In conclusion, the implemented solution effectively manages project data for XYZ Company. It utilizes a list-based approach to store data about ongoing, on-hold, and completed projects, along with the number of available workers. The code provides functions to handle various project management tasks, including adding, removing, updating, and viewing project information. The code incorporates input validation and error handling to ensure data integrity and a smooth user experience. Overall, the program is implemented to meet and effectively addresses the project management requirements for XYZ Company.