Assignment 2

Calendar

Done by:

Elaf Abdullah Saleh Alhaddad 31063977
Sutulova Tatiana 30806151

WhiteBox testing **3**
1.1 Displaying events in the past 5 years and displaying events in the upcoming two years 3
1.1.1: Testing method: Statement coverage 3
1.2 Navigating through different years, months and days 4
1.2.1: Testing method: Condition coverage and branch coverage 4
1.3 Searching an event by the keyword 4
1.3.1: Printing events. Testing method : MC/DC 4
1.3.2: Printing reminders. Branch coverage. 6
1.5 Deleting Events and reminders 6
1.2.1: Testing method: Condition coverage and branch coverage 6
1.5 Mocking 8
1.5.1: Mocking the API 8
1.5.2: Mocking the output 8
1.5.3: Mocking other functions 8
1.5.4: Mocking the inputs 9
1.6 Coverage 9

BlackBox testing **9**
2.1 Get events and reminders in the past 5 years 9
2.2 Get events and reminders in the future 10
2.3 Navigating through different years, months and days 11
2.4 Get event by a keyword 19
2.5 Delete event/reminder functionality 21
2.6 Main menu functionality 26

1. WhiteBox testing

1.1 Displaying events in the past 5 years and displaying events in the upcoming two years

1.1.1: Testing method: Statement coverage

The user can use these functionalities by choosing them from the menu that is displayed as soon as the app is activated. All these functionalities have one path that the system runs through when they are called, hence we have chosen to test these functionalities using statement coverage.

1.2 Navigating through different years, months and days

1.2.1: Testing method: Condition coverage and branch coverage

In this functionality, the user gets to choose the period of time they want the events to be listed. Branch coverage has been chosen to test the following functionalities and the CFG representing all the branches being tested is displayed in *get_event_by_timeline.pdf* in the CFG folder. The user chooses the time range within which all the events will be shown and if there are events, the user is able to choose one of them by inputting the title, which will represent the detailed information about the chosen event and allow it to be deleted.

1.3 Searching an event by the keyword

1.3.1: Printing events. Testing method : MC/DC

MC/DC testing is chosen to test the following functionality, which allows the user to search the event by typing the key word. All the branches are shown in the document named *“get _event_by_keyword.pdf”* and the testing is implemented to test whether the event is found properly after the user inputs the keyword.
Outcome 1= if (subtype=="" and event[type] == key_word) or (subtype!="" and event[subtype][type] == key_word)

A = subtype==""
B = event[type] == key_word
C= event[subtype][type] == key_word

Outcome1= If (A and B) or (not A and C)

Test	**A**	**B**	**C**	**Outcome 1**
1	T	T	T	T
2	T	T	F	T
3	T	F	T	F
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F

7	F	F	T	T
8	F	F	F	F

For A: {2,6}; {3,7}
For B: {1,3}; {2,4}
For C: {5,6}; {7,8}

The MC/DC: 2,4,5,6 is found by following the steps to identify the 'important' combinations, that reduces the test cases needed to test the following test frame with the 100% path coverage . However, logically line 5 (as well as line 1) cannot be tested since conditions B (event[type] == key_word) and C(event[subtype][type] == key_word), are mutually exclusive (key_word cannot equal to both event[type] and event[subtype][type]) .Thus, line 5 has to be excluded from the table.
New MC/DC: 2,4,6,7,8 , which is not following N+1, where N is the number of conditions, however it suits the limit which is N*2 and gives the logical testing with the 100% MC/DC coverage.

1.3.2: Printing reminders. Branch coverage.

Reminders are printed along with the rest of the information about the existing events, which is used in all the functionalities mentioned in the menu of the app. The main condition to find the reminder of an event is that the following event exists in the database.
There are three possible conditions affecting the reminder outcome:

1. The reminder is *default google reminder*
2. There is *no reminder* for the chosen event
3. Reminder is a *manually created* by the user reminder

In order to test all the possible outcomes, the branch coverage method is used, since all the decisions mentioned before have two possible outcomes, which is the main rule for the branch coverage testing. The control flow graph(CFG), which represents all the branches being tested can be found in the CFG folder with the *“represent_event_reminders.pdf”* name.

1.5 Deleting Events and reminders

1.2.1: Testing method: Condition coverage and branch coverage

This method is only called when selecting a specific event by using the search functionality or selecting it after using the navigation through different timeline functionality.

Conditional coverage will be used to ensure that the application outlines the possible inputs that the user can provide based on the selected event. For example, a user will not be able to delete a reminder for an event if the event doesn’t have a reminder. The possible inputs will be tested using branch coverage.

The tests that will be conducted are outlined by the table below and more detailed representation of branching can be found in the CFG diagram named *“delete_event_reminder.pdf “* in a CFG folder.

Conditions:

A - Does the event use the google default reminder
B - Does the event have a customised reminder

Outcomes:

Outcome 1 - the ability to delete event/ reminders
Outcome 2 - the ability to delete the event only

Test	A	B	Outcome 1	Outcome 2
1	T	F	T	F
2	F	F	F	T
3	F	T	T	F

* The case when both conditions are true is excluded since conditions are mutually exclusive.

1.5 Mocking

Mocking is used in unit testing, when the object under test has dependencies on other complex objects, so in order to test this object in isolation, other objects may be replaced by mocks that simulate the behavior of the real objects. We have used mocking strategies for several purposes:

1.5.1: Mocking the API

mock_api= Mock() and *mock_api=MagicMock()* are used to mock the google calendar API, which is what the entire application functionality is based on. Since the amount of time that it takes to access the API is approximately 1 second, it will be inefficient, when numerous tests are implemented. Difference between Mock() and MagicMock() is that MagicMock() is iterable and therefore it allows to iterate through the events in the mocked API.

1.5.2: Mocking the output

@patch('builtins.print') allows you to mock the output, which is needed to test what is printed to the console. Since the Calendar app is based on the interaction of the user with the I/O console, it allows to test whether the output shown to the user is right.

1.5.3: Mocking other functions

@patch('Calendar.Calendar.delete_event_reminder') and *@patch('Calendar.Calendar.get_event_by_keyword')* are responsible for patching the outputs of the functions that are called in the function being tested by setting their return_value="". This is needed to make sure that these side outputs do not affect the outcome of the function that is being tested.

1.5.4: Mocking the inputs

@patch('builtins.input', side_effect=['2020', '1', 'End Year']) is used to patch the inputs that the user is prompted for in the function that is being tested. side_effect is an array, that contains the values that will be mocking the user inputs in the same order (e.g *first prompt->2020*, etc)

1.6 Coverage

The total coverage of the app by the testing methods is less than 100%, which is explained by the fact that significant percent of the total code for the app is the menu function that we decided not to test since it does not represent the calendar app requirement. Moreover, the function that interacts directly with the api (get_calendar_api) was not considered for testing.

2. BlackBox testing

2.1 Get events and reminders in the past 5 years

Test Name:	Events & reminders in the past 5 years	**Number of test cases:**	1
Description:	This test suite includes a test case that is examining if the application properly returns events and reminders in the past 5 years	**Tester's information:**	Elaf Abdullah
Rationale	The strategy used in this test suite is random testing, since there is no variety of input the user can suggest in order to get the required information.		

Test case #001			**Description:** When the corresponding number is chosen in the main menu, the app is supposed to return all the events and reminders in next2 years.	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 1 in the main menu	The app returns all the events and their reminders in the past 5 years	The app returns all the events and their reminders in the past 5 years	Pass

2.2 Get events and reminders in the future

Test Name:	Events & reminders in the future	**Number of test cases:**	1
Description:	This test suite includes a test case that is examining if the application properly returns events and reminders in next two years	**Tester's information:**	Tatiana Sutulova
Rationale	The strategy used in this test suite is random testing, since there is no variety of input the user can suggest in order to get the required information.		

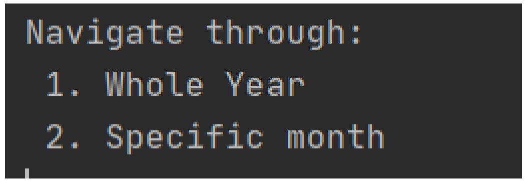
Test case #002			**Description:** When the corresponding number is chosen in the main menu, the app is supposed to return all the events and reminders in next2 years.	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 2 in the main menu	The app returns all the events and reminders in next two years	The app returns all the events and reminders in next two years	Pass

2.3 Navigating through different years, months and days

Test Name:	Navigating through years, months and days	**Number of test cases:**	
Description:	This test suite includes test cases that are examining if the user is able to navigate through the events in the calendar based on the timer period they have chosen	**Tester's information:**	Elaf Abdullah
Rationale	The strategy used in this test suite is category testing, where categories are - Getting events from a specific year (whole year) - Getting events from a specific month (from a specific year) - Valid inputs: 1~ 12 based on the 12 months a year - Invalid inputs: smaller than 1 and greater than 12 - Getting events from a specific date - Valid inputs: dates that are present within a month based on the yearly calendar - Invalid inputs: dates that are not present within a month (e.g. 31/11/2020)		

Test case #003	**Description:** Getting the events from a specific year. Due to the possibility of many different inputs, we use equivalence testing with the assumption that if the test case passes for one specific year, it will pass for all other years in the calendar. Assumptions for this test: - This test is conducted based on the assumption that there are events present in the chosen year. - The delete event functionality is working as expected

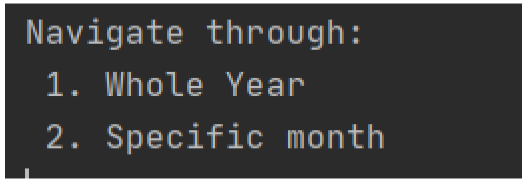
Step	Test Step/Input	Expected outputs	Actual outputs	Pass/fail
1	User input 3 in the main menu	The app asks for a year that user would like to navigate through	The app asks for a year that user would like to navigate through	Pass

2	User input the year to navigate through	The app asks the user whether they would like to navigate through the whole year or a specific month	
			```
			Navigate through:
			 1. Whole Year
			 2. Specific month
			```

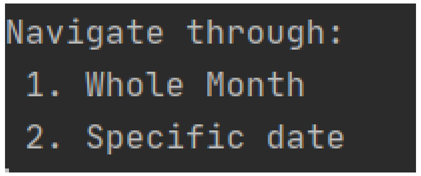
3	User input 1 to navigate through whole year	All events in the chosen year are listed and the user is prompted to input a title to select a specific event	All events in the chosen year are listed and the user is prompted to input a title to select a specific event	
4	User inputs the title for the event	Details of the selected event is printed out with an option to delete the event	Details of the selected event is printed out with an option to delete the event	

Test case #004	**Description:** Getting the events from a specific month in a specific year. Due to the possibility of many different inputs, we use equivalence testing with the assumption that if the test case passes for one specific month, it will pass for all other months in the calendar. Assumptions for this test: - This test is conducted based on the assumption that there are events present in the chosen month. - This test uses valid inputs - The delete event functionality is working as expected

Step	Test Step/Input	Expected outputs	Actual outputs	Pass/fail
1	User input 3 in the main menu	The app asks for a year that user would like to navigate through	The app asks for a year that user would like to navigate through	Pass

2	User input the year to navigate through	The app asks the user whether they would like to navigate through the whole year or a specific month	
			```
			Navigate through:
			 1. Whole Year
			 2. Specific month
			```

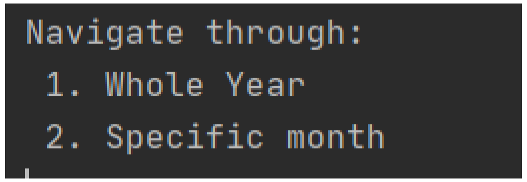
3	User input 2 to navigate through a specific month	App prompt user to input the month they want to navigate through	App prompt user to input the month they want to navigate through	

4	User input the month to navigate through	The app asks the user whether the would like to navigate through the whole month in that year or a specific date	
			```
			Navigate through:
			 1. Whole Month
			 2. Specific date
			```

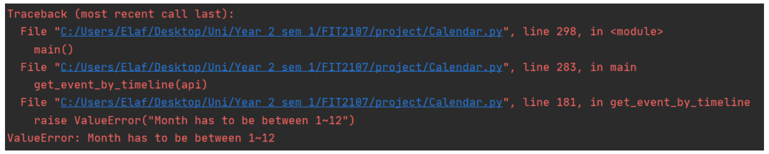
5	User input 1 to navigate through whole month	All events in the chosen month of the year are listed and the user is prompted to input a title to select a specific event	All events in the chosen month of the year are listed and the user is prompted to input a title to select a specific event	
6	User inputs the title for the event	Details of the selected event is printed out with an option to delete the event	Details of the selected event is printed out with an option to delete the event	

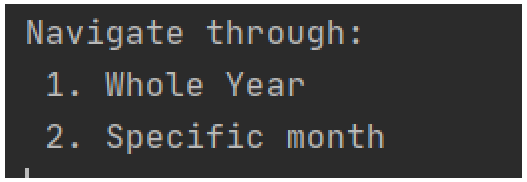
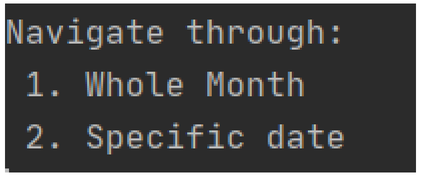
Test case #005	**Description:** Getting the events from a specific month in a specific year. Due to the possibility of many different inputs, we

			use equivalence testing with the assumption that if the test case passes for one specific month, it will pass for all other months in the calendar. Assumptions for this test: - This test is conducted based on the assumption that there are events present in the chosen month. - This test uses invalid inputs - The delete event functionality is working as expected	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 3 in the main menu	The app asks for a year that user would like to navigate through	The app asks for a year that user would like to navigate through	Pass

2	User input the year to navigate through	The app asks the user whether they would like to navigate through the whole year or a specific month	
			`Navigate through:`
			`  1. Whole Year`
			`  2. Specific month`	

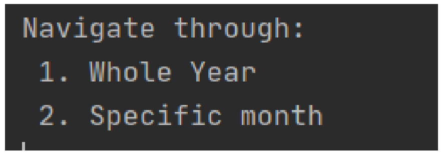
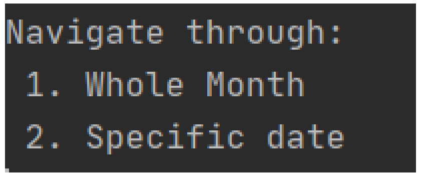
3	User input 2 to navigate through a specific month	App prompt user to input the month they want to navigate through	App prompt user to input the month they want to navigate through	

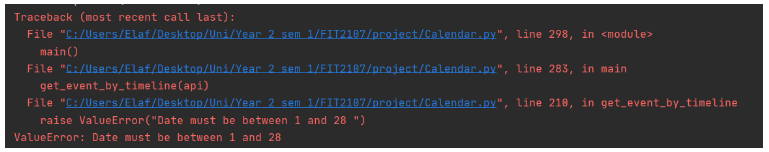
4	User input invalid month to navigate through (smaller than 1 or greater than 12)	Error message is returned	
			`Traceback (most recent call last):`
			`  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 298, in <module>`
			`    main()`
			`  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 283, in main`
			`    get_event_by_timeline(api)`
			`  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 181, in get_event_by_timeline`
			`    raise ValueError("Month has to be between 1~12")`
			`ValueError: Month has to be between 1~12`	

Test case #006			**Description:** Getting the events from a specific date. Due to the possibility of many different inputs, we use equivalence testing with the assumption that if the test case passes for one specific date, it will pass for all other dates in the calendar. Assumptions for this test: - This test is conducted based on the assumption that there are events present in the chosen date - This test uses valid inputs - The delete event functionality is working as expected	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 3 in the main menu	The app asks for a year that user would like to navigate through	The app asks for a year that user would like to navigate through	Pass
2	User input the year to navigate through	The app asks the user whether they would like to navigate through the whole year or a specific month	 Navigate through: 1. Whole Year 2. Specific month	
3	User input 2 to navigate through a specific month	App prompt user to input the month they want to navigate through	App prompt user to input the month they want to navigate through	
4	User input the month to navigate through	The app asks the user whether the would like to navigate through the whole month in that year or a specific date	 Navigate through: 1. Whole Month 2. Specific date	

5	User input 2 to navigate through a specific date	App promote user to input a specific date based on the month they have chosen : - 1~31 if the month they have chosen is Jan/Mar/May/July/Aug/Oct/Dec - 1~30 if the month they have chosen is Apr/July/Sept/Nov - 1~29 if the month they have chosen is Feb in a leap year - 1~28 if the month they have chosen is Feb in any year that isn't a leap year	App promote user to input a specific date based on the month they have chosen : - 1~31 if the month they have chosen is Jan/Mar/May/July/Aug/Oct/Dec - 1~30 if the month they have chosen is Apr/July/Sept/Nov - 1~29 if the month they have chosen is Feb in a leap year - 1~28 if the month they have chosen is Feb in any year that isn't a leap year	
6	User input the valid date	All events in the chosen date are listed and the user is prompted to input a title to select a specific event	All events in the chosen date are listed and the user is prompted to input a title to select a specific event	
7	User inputs the title for the event	Details of the selected event is printed out with an option to delete the event	Details of the selected event is printed out with an option to delete the event	

Test case #007			**Description:** Getting the events from a specific date. Due to the possibility of many different inputs, we use equivalence testing with the assumption that if the test case passes for one specific date, it will pass for all other dates in the calendar. Assumptions for this test: - This test is conducted based on the assumption that there are events present in the chosen date - This test uses invalid inputs - The delete event functionality is working as expected	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**

1	User input 3 in the main menu	The app asks for a year that user would like to navigate through	The app asks for a year that user would like to navigate through	Pass
2	User input the year to navigate through	The app asks the user whether they would like to navigate through the whole year or a specific month	 `Navigate through:` `1. Whole Year` `2. Specific month`	
3	User input 2 to navigate through a specific month	App prompt user to input the month they want to navigate through	App prompt user to input the month they want to navigate through	
4	User input the month to navigate through	The app asks the user whether the would like to navigate through the whole month in that year or a specific date	 `Navigate through:` `1. Whole Month` `2. Specific date`	
5	User input 2 to navigate through a specific date	App promote user to input a specific date based on the month they have chosen : - 1~31 if the month they have chosen is Jan/Mar/May/July/Aug/Oct/Dec - 1~30 if the month they have chosen is Apr/July/Sept/Nov - 1~29 if the month they have chosen is Feb in a leap year - 1~28 if the month they have chosen is Feb in any year that isn't a leap year	App promote user to input a specific date based on the month they have chosen : - 1~31 if the month they have chosen is Jan/Mar/May/July/Aug/Oct/Dec - 1~30 if the month they have chosen is Apr/July/Sept/Nov - 1~29 if the month they have chosen is Feb in a leap year - 1~28 if the month they have chosen is Feb in any year that isn't a leap year	

6	User input the invalid date	Error message is returned		

```
Traceback (most recent call last):
  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 298, in <module>
    main()
  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 283, in main
    get_event_by_timeline(api)
  File "C:/Users/Elaf/Desktop/Uni/Year 2 sem 1/FIT2107/project/Calendar.py", line 210, in get_event_by_timeline
    raise ValueError("Date must be between 1 and 28 ")
ValueError: Date must be between 1 and 28
```

2.4 Get event by a keyword

Test Name:	Getting event by a keyword	**Number of test cases:**	3
Description:	This test suite includes test cases that are examining if the user is able to find an event by inputting a keyword.	**Tester's information:**	Tatiana Sutulova
Rationale	The strategy used in this test suite is category testing, where categories are : - valid input: integer in the range from 1 to 4 - valid input: the name of the functionality - invalid input: any random input, which is not included in first two categories		

Test case #008			**Description:** Inputting number 2, which corresponds to the title of the event, which is supposed to find all the events with the following title.	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**

1	User input 4 in the main menu	The app returns all the events with the following title one by one asking if the user wants to delete them	The app returns all the events with the following title one by one asking if the user wants to delete them	Pass
2	User inputs 2, when the app asks what kind of the keyword the user want to use			
3	User inputs the proper event title			

Test case #009			**Description:** Inputting “Location”, as the type of the keyword	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 4 in the main menu	The app returns all the events with the following location one by one asking if the user wants to delete them	The app returns all the events with the following location one by one asking if the user wants to delete them	Pass
2	User inputs “Location”, when the app asks what kind of the keyword the user want to use			
3	User inputs the proper location			

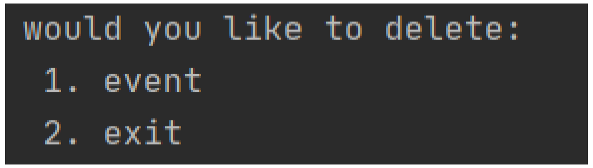
Test case #010			**Description:** Inputting random value, as the type of the keyword	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User inputs 4 in the main menu	The app returns the message notifying the user about the incorrect input and allows the user to choose the kind of the keyword again.	The app returns the message notifying the user about the incorrect input and allows the user to choose the kind of the keyword again.	Pass
2	User inputs 7, when the app asks what kind of the keyword the user want to use			

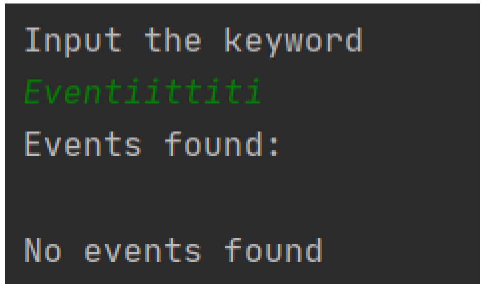
2.5 Delete event/reminder functionality

Test Name:	Deleting an event or reminder	**Number of test cases:**	1
Description:	This test suite includes test case that are examining if the user will be able to delete an event	**Tester's information:**	Elaf Abdullah
Rationale	The strategy used in this test suite is category testing, where the categories are: - Event has a reminder - Deleting event - Deleting reminder - Exit - Event has no reminder - Deleting event		

	- Exit Since the process for deleting the event and exiting is the same for both categories, we assume that if the tests pass for the “event has no reminder” category then they will pass with the other category as well.

Test case #011			**Description:** This test case will test whether the delete event functionality works as expected for when event has no reminder Assumptions: - The user can only delete a specific event after accessing it using the Search by keyword functionality or navigating through years/months/dates functionality - This test is conducted based on the assumption that search by keyword and navigation through years/months/dates functionality work as expected - If the test results will be the same for when the event has a reminder	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**

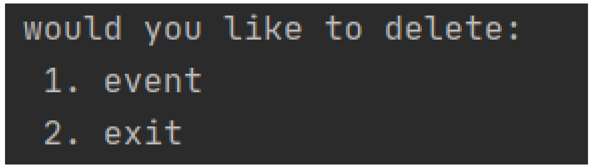
1	User select the event they would like to delete using the search by keyword functionality/ navigation through years/months/days functionality	The app prompts the user if they would like to delete the event	
			```
			would you like to delete:
			 1. event
			 2. exit
			```
				Pass

2	User input 1 to delete the event	Event gets deleted, this can be tested by printing out the events/searching for the deleted event to check if it still exists in the system	Event gets deleted. 
			```
			Input the keyword
			Eventiittiti
			Events found:

			No events found
			```

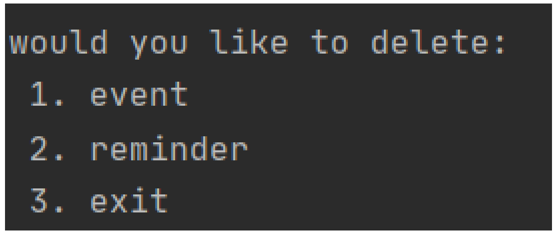
Test case #012	**Description:** This test case will test whether the user can exit and not delete an event for when event has no reminder Assumptions: - The user can only delete a specific event after accessing it using the Search by keyword functionality or navigating through years/months/dates functionality - This test is conducted based on the assumption that search by keyword and navigation through

			- years/months/dates functionality work as expected - If the test results will be the same for when the event has a reminder	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**

1	User select the event they would like to delete using the search by keyword functionality/ navigation through years/months/days functionality	The app prompts the user if they would like to delete the event	
			`would you like to delete:`
			`1. event`
			`2. exit`
				Pass

2	User input 2 to exit	User exits and is taken back to the main menu	User exits and is taken back to the main menu	

Test case #012	**Description:** This test case will test whether the user can delete a reminder Assumptions: - The user can only delete a specific event after accessing it using the Search by keyword functionality or navigating through years/months/dates functionality - This test is conducted based on the assumption that search by keyword and navigation through years/months/dates functionality work as expected

Step	Test Step/Input	Expected outputs	Actual outputs	Pass/fail
1	User select the event they would like to delete using the search by keyword functionality/ navigation through years/months/days functionality	The app prompts the user if they would like to delete the reminder	 `would you like to delete:` `1. event` `2. reminder` `3. exit`	Pass
2	User input 2 to delete the reminder	Reminder for the event gets deleted, this can be tested by printing out the events/searching for the event to check if it still has a reminder	Reminder gets deleted and application shows that event has no reminder.	

2.6 Main menu functionality

Test Name:	Accessing different functionality from the main menu	**Number of test cases:**	3
Description:	This test suite includes test cases that are examining if the user is able to properly access any functionality of the application from the main menu	**Tester's information:**	Tatiana Sutulova
Rationale	The strategy used in this test suite is category testing, where categories are - valid input: integer in the range from 1 to 4 - valid input: 5, that allows the user to exit the program - invalid input: any random input, which is not included in first two categories		

Test case #012			**Description:** Inputting number 1, which corresponds to show events in last 5 years	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User input 1 in the main menu	The app returns all the events in last 5 years	The app returns all the events in last 5 years	Pass

Test case #013			**Description:** Inputting 5 in the main menu, which corresponds to exiting the program	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**

1	User input 5 in the main menu	The app stops an execution	The app stops an execution	Pass

Test case #014			**Description:** Inputting 10 in the main menu, which is an invalid input	
Step	**Test Step/Input**	**Expected outputs**	**Actual outputs**	**Pass/fail**
1	User inputs 10 in the main menu	The app returns the corresponding message letting the user know that the input is invalid and allows the user to input it again till the user chooses the proper value	The app returns the corresponding message letting the user know that the input is invalid and allows the user to input it again till the user chooses the proper value	Fail