CS213: Object Oriented Programming

Cairo University, Faculty of Computers
and Artificial Intelligence

FACULTY OF COMPUTERS AND INFROMATION,
CAIRO UNIVERSITY

CS213: Programming II

Year 2023-2024

First Semester

Assignment 2 - Project description

Project Team

Course Instructors:

Dr. Mohammad El-Ramly.

Team Work:

Name	**ID**	**Work**
Salma Gamal Kamel Mahmed Radwan	2022 1073	Task:1 2,5,8,11. Task :2 bool operator< bool operator> bool operator== friend ostream& operator <<
Malak Ahmed	2022 1157	Task:1 3,6,9,12 Task:2 Operator +() Operator –()
Nadra Mahmoud Saad	2022 0355	Task:1 1,4,7,10 Task:2 Is valid function Defult constructor Parametrize Constructor Copy Constructor Assignment Operator Setter function Size function Sign function Print function

<u>**Task1**</u>:

Problem 1:

in this question to handle the sentences we first declare a contener to input and output and tack the user sentence and put it in input contener and loop on it

If their is two consecutive spaces do nothing

If the char is upperbound make it lower

Ten make the first char upperbound then print the output till the dot.

Problem 2:

1. Create a function called `Replace_Pronouns` that takes a string `s` as input.

2. Define two arrays:`search_word` ,`replace_word'

3. Replace the substring "he" with "he or she" in the input string `s`.

4. Loop through the pronoun arrays using a `for` loop.

Problem 3:

first make a vector then loop over the string if i found the delimiter then add the text before it in an index of a the vector and jump to the index after the delimiter.

Problem 4:

In this question, we want to print prime numbers from 1 to a specific number, so using some simple mathematical operations and looping on the numbers, we will remove the number that does not meet the condition, and after completion, we print the new system.

Problem 5:

1. Define a struct called “Player” that contains a player’s name and score.

2. Define a function called “compare” that compares two players based on their scores.

3. Define a vector called “playerList” to store a list of players.

4. Enter an infinite loop using the “while (true)” statement.

5. Display a menu with four options for the user to choose .

6. Use a switch statement to perform different actions based on the user’s choice.

- If the user chooses option 1, prompt the user to enter a player’s name and score.

  - If the size of the playerList is less than 10, add the new player to the list.

  - Otherwise, find the player with the lowest score in the list using the “min_element” function and compare function, and replace it with the new player if the new player has a higher score.

- If the user chooses option 2, sort the playerList in descending order based on the scores using the “sort” function and the compare function. Then, print the top 10 players’ names and scores.

- If the user chooses option 3, prompt the user to enter a player’s name. Iterate over the playerList and find the player with the matching name. Track the highest score found for that player. If the player is found, print their highest score. Otherwise, print a message indicating that the player’s name has not been input or is not in the top 10.

- If the user chooses option 4, exit the program by returning 0.

- If the user chooses an invalid option, display an error message.

7. The loop continues to display the menu and process the user’s choices until the program is exited.

Problem 6:

Binary number: the binary of a number is the continuous results from (number%2).

Number: first i want to get the range of k by k^2 then loop on that range and add the corresponding binary number to the wanted one.

Problem 7:

In this question, we use the principles of backtracing and recursion to print domino pieces placed in an order where each piece in the specified number has a piece next to it if the two numbers match. We did this through two functions. Every time the for loop is entered in the first function(front_domino), an independent series of events occurs and in the solution function (solve),We go through three stages until the solution is printed if it exists, and nothing is printed if there is no solution.

Problem 8:

1. Define a function called “Pattern” that takes two parameters: “n” (the number of stars) and “i” (the number of spaces).

2. Check if “n” is equal to 1. If it is, it means we have reached the base case where we need to print a single star.

   - In this case, iterate “i” times and print a space.

   - Then, print a single star.

3. If “n” is not equal to 1, it means we need to print a fractal pattern.

   - Recursively call the “Pattern” function with “n/2” stars and the same number of spaces “i”. This will print the top half of the pattern.

   - Iterate “i” times and print a space.

   - Iterate “n” times and print a star.

   - Move to the next line.

   - Recursively call the “Pattern” function with “n/2” stars and “i+n/2” spaces. This will print the bottom half of the pattern

The “Pattern” function uses recursion to divide the pattern into smaller parts until it reaches the base case of printing a single star. Then, it builds the pattern by printing the top half, followed by a line of stars, and then the bottom half. The main function takes user input for the number of stars and spaces and calls the “Pattern” function to generate the fractal pattern.

Problem 9:

The concept is to check the possible solution of all solutions but i add a vector of a bool to make sure that the program wont inter an infinite loop in some cases like (100).

Problem 10:

In this question, we have a file for the inputs, a file for the outputs, and a file for the list containing the prohibited words and their alternatives. We use the map to contain the forbidden words as a key and their alternatives as a value. After that, looping over the inputs, and every time we find a forbidden word, I replace it with the alternative, and then print the new sentence in the output file.

Problem 11:

1. Define a function called “Compare_characterBycharacter” that takes two parameters: “file1” and “file2” (both are ifstream objects representing two input files).

2. Initialize variables “ch1” and “ch2” to store characters read from each file, and “lineCount” and “charCount” to keep track of the current line and character positions.

3. Use a while loop to read characters from both files character by character until either file reaches the end.

   - Compare the characters read from the two files. If they are not equal, print the line and character position where the files differ and return false.

   - If the character read is a newline character (‘\n’), increment the lineCount and reset the charCount to 0.

4. After the while loop, check if both files have reached the end of file (EOF). If they have, print a message that the files are identical and return true. Otherwise, print a message that the files have different lengths and return false.

5. Define a function called “Compare_wordByword” that takes two parameters: “file1” and “file2” (both are ifstream objects representing two input files).

6. Initialize variables “word1” and “word2” to store words read from each file, and “lineCount” to keep track of the current line position.

7. Use a while loop to read words from both files word by word until either file reaches the end.

   - Compare the words read from the two files. If they are not equal, print the line and the mismatched words and return false.

8. After the while loop, check if both files have reached the end of file (EOF). If they have, print a message that the files are identical and return true. Otherwise, print a message that the files have different lengths and return false.

9. In the main function:

- Read the filenames of the two files from the user.
- Read the comparison type (“character” or “word”) from the user.
- Open the two files using the filenames provided by the user.
- Check if the files were opened successfully. If not, print an error message and return 1 to indicate an error.
- Call the appropriate comparison function based on the comparison type provided by the user.
- Close the files.
- Return 0 to indicate successful program execution.

Problem 12:

make a 2 vectors of words the most frequented in phishing emails and their corresponding rank and then get the file word by word and turn its word to all small cases (to make it more efficient) and then check how many times the phishing words frequented.

<u>**Task 2**</u>:

-BigReal class .

Function members :

Addition : first make the size the same to make it easier to add or subtract then check for the sign to determine the sign of the result then add the corresponding index in the two numbers and put in the stack and then transfer the stack to the result.

Subtract: first make the size the same to make it easier to add or subtract then check for the sign to determine the sign of the result then subtract the corresponding index in the two numbers and put in the stack and then transfer the stack to the result.

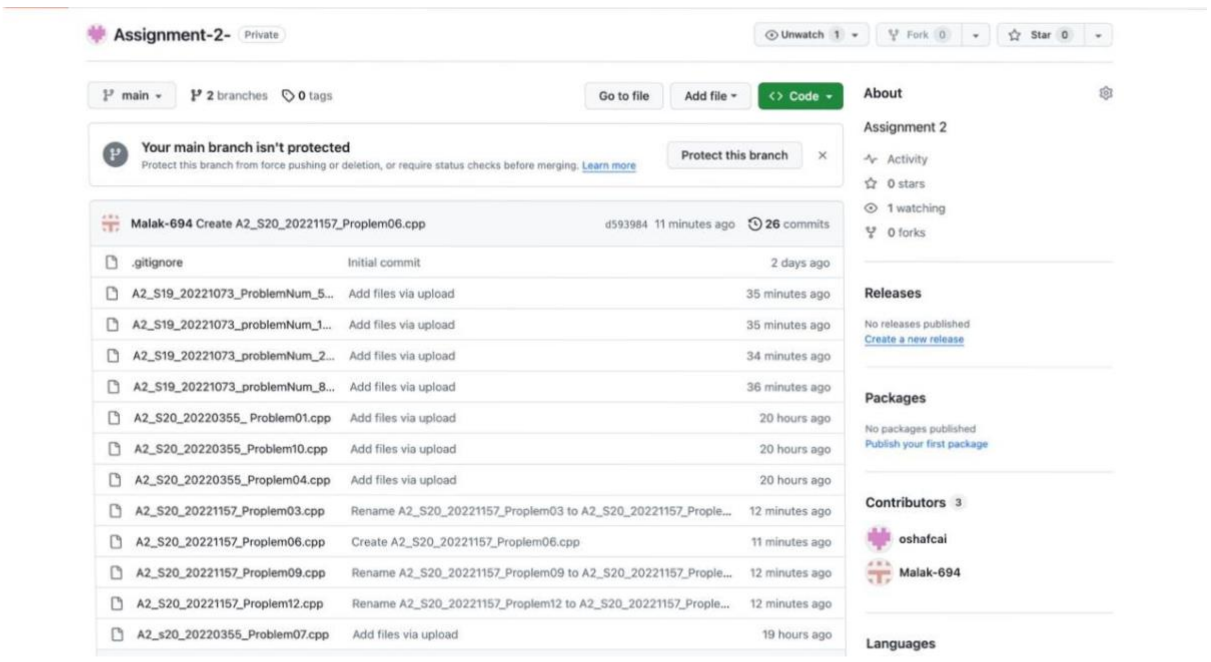
bool Bigreal::operator<(Bigreal anotherReal)`: This function overloads the less than operator (`<`) for the `Bigreal` class. It compares two `Bigreal` objects by converting their integer and fraction parts to double values using the `stod()` function and checks if the first object is less than the second object. It returns `true` if the first object is less than the second object, and `false` otherwise.

bool Bigreal::operator>(Bigreal anotherReal)`: This function overloads the greater than operator (`>`) for the `Bigreal` class. It compares two `Bigreal` objects by converting their integer and fraction parts to double values using the `stod()`

function and checks if the first object is greater than the second object. It returns `true` if the first object is greater than the second object, and `false` otherwise.

IsValidReal: I check for the sign of the number , if there 2 sign since it is invalid (result 1) and in the same tome i check if there more than 1 digit it is invalid number (result 2) and the final destion wether it is valid or not is the (and &&) of the two results.

<u>GitHub work</u>:



Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. Learn more
Protect this branch

Malak-694 Create A2_S20_20221157_Proplem06.cpp d593984 11 minutes ago 26 commits

.gitignore	Initial commit	2 days ago
A2_S19_20221073_ProblemNum_5...	Add files via upload	35 minutes ago
A2_S19_20221073_problemNum_1...	Add files via upload	35 minutes ago
A2_S19_20221073_problemNum_2...	Add files via upload	34 minutes ago
A2_S19_20221073_problemNum_8...	Add files via upload	36 minutes ago
A2_S20_20220355_ Problem01.cpp	Add files via upload	20 hours ago
A2_S20_20220355_Problem10.cpp	Add files via upload	20 hours ago
A2_S20_20220355_Proplem04.cpp	Add files via upload	20 hours ago
A2_S20_20221157_Proplem03.cpp	Rename A2_S20_20221157_Proplem03 to A2_S20_20221157_Prople...	12 minutes ago
A2_S20_20221157_Proplem06.cpp	Create A2_S20_20221157_Proplem06.cpp	11 minutes ago
A2_S20_20221157_Proplem09.cpp	Rename A2_S20_20221157_Proplem09 to A2_S20_20221157_Prople...	12 minutes ago
A2_S20_20221157_Proplem12.cpp	Rename A2_S20_20221157_Proplem12 to A2_S20_20221157_Prople...	12 minutes ago
A2_s20_20220355_Problem07.cpp	Add files via upload	19 hours ago