Name: Umut kuruluk
No:210101118

Department: Computer Engineering

Regular Expression Parser

1-)Introduction:

-The project is Regular Expression Parser, with the input you give as a Regular expression program takes and try to find it inside of a specific paragraph.

2-)Problem Definition:

-Being able to search for multiple words by typing 1 Regular expression instead of trying them one by one.

3-)Pseudocode:

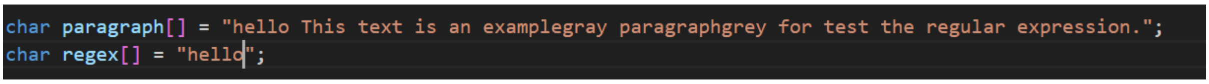
```
class Main:
    method paragraph_contains(paragraph_arr, last_arr):
        last_arr.add(" ")
        check_word = []
        for i in range(last_arr.length()):
            if last_arr[i] == " ":
                for j in range(paragraph_arr.length()):
                    for k in range(check_word.length()):
```

```
                    if check_word[k] != paragraph_arr[j+k]:
                        break
                    if k == check_word.length() - 1:
                        return True
            check_word.clear()
            continue
        check_word.add(last_arr[i])
    return False

method regex_parentheses(regex_arr, last_arr):
    before_parantheses = []
    after_parantheses = []
    onhold = []
    for i in range(regex_arr.length()):
        if regex_arr[i] == "(":
            break
        before_parantheses.add(regex_arr[i])
    for i in range(regex_arr.length()):
        if regex_arr[i] == ")":
            for j in range(i + 1, regex_arr.length()):
                after_parantheses.add(regex_arr[j])
            break
    for i in range(regex_arr.length()):
        if regex_arr[i] == "(":
            for j in range(i + 1, regex_arr.length()):
                if regex_arr[j] == "|" or regex_arr[j] == ")":
                    last_arr.add(" ")
                    onhold.clear()
                    break
                onhold.add(regex_arr[j])
                for k in range(before_parantheses.length()):
                    last_arr.add(before_parantheses[k])
                for k in range(onhold.length()):
                    last_arr.add(onhold[k])
                for k in range(after_parantheses.length()):
                    last_arr.add(after_parantheses[k])
                last_arr.add(" ")
                onhold.clear()

method regex_brackets(regex_arr, last_arr):
    // Pseudo-code for handling brackets

method regex_questionmark(regex_arr, last_arr):
    // Pseudo-code for handling question marks
method regex_star(regex_arr, last_arr, paragraph_length):
    // Pseudo-code for handling asterisks

method regex_sum(regex_arr, last_arr, paragraph_length):
    // Pseudo-code for handling plus signs

method regex_number(regex_arr, last_arr):
    // Pseudo-code for handling numbers

method regex_curly(regex_arr, last_arr, paragraph_length):
    // Pseudo-code for handling curly braces

method regex_OR(regex_arr, last_arr):
    // Pseudo-code for handling OR operator

method regex_check(paragraph, regex, last_arr):
    paragraph_arr = paragraph.split("")
    regex_arr = regex.split("")

    for i in range(regex.length()):
        regex_char = regex_arr[i]
        if regex_char == "(":
```

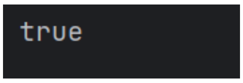
```
            regex_parentheses(regex_arr, last_arr)
        if regex_char == "[":
            regex_brackets(regex_arr, last_arr)
        // Similar calls for other regex characters
    return paragraph_contains(paragraph_arr, last_arr)

method main():
    last_arr = []
    paragraph = "This text is an example paragraph for test the regular expression."
    regex = "text|test"
    res = regex_check(paragraph, regex, last_arr)
    print(res)
```

4-)Demo:

1-

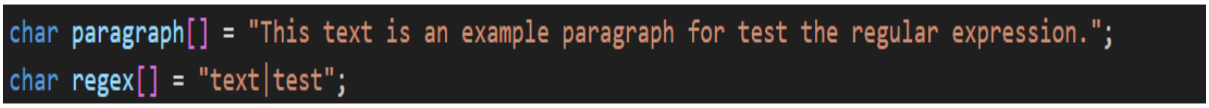


```
char paragraph[] = "hello This text is an examplegray paragraphgrey for test the regular expression.";
char regex[] = "hello";
```

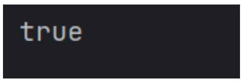


```
true
```

2-

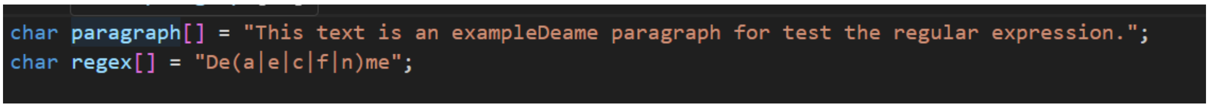


```
char paragraph[] = "This text is an example paragraph for test the regular expression.";
char regex[] = "text|test";
```

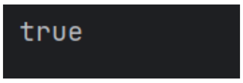


```
true
```

3-

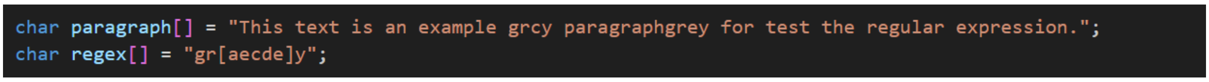


```
char paragraph[] = "This text is an exampleDeame paragraph for test the regular expression.";
char regex[] = "De(a|e|c|f|n)me";
```

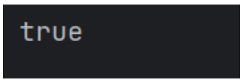


```
true
```

4-

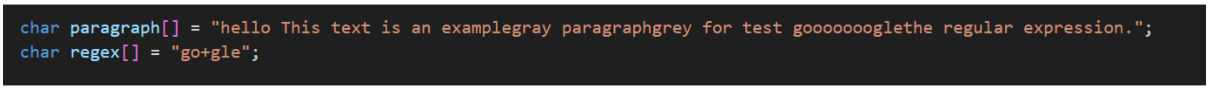


```
char paragraph[] = "This text is an example grcy paragraphgrey for test the regular expression.";
char regex[] = "gr[aecde]y";
```

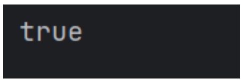


```
true
```

5-

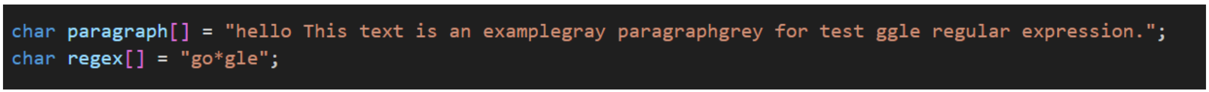


```
char paragraph[] = "hello This text is an examplegray paragraphgrey for test goooooooglethe regular expression.";
char regex[] = "go+gle";
```

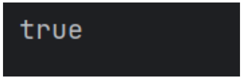


```
true
```

6-

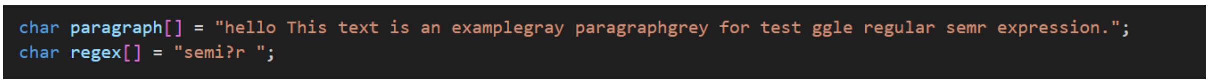


```
char paragraph[] = "hello This text is an examplegray paragraphgrey for test ggle regular expression.";
char regex[] = "go*gle";
```

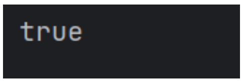


```
true
```

7-

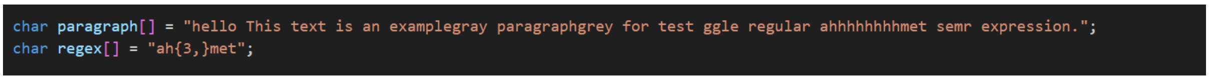


```
char paragraph[] = "hello This text is an examplegray paragraphgrey for test ggle regular semr expression.";
char regex[] = "semi?r ";
```

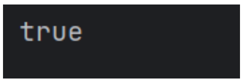


```
true
```

8-



```
char paragraph[] = "hello This text is an examplegray paragraphgrey for test ggle regular ahhhhhhhhhmet semr expression.";
char regex[] = "ah{3,}met";
```



```
true
```

5-)References:

https://regexr.com/

https://youtu.be/rhzKDrUiJVk?si=Wc6kMgfH1cUQQcQg