Muneel Haider Assignment 3

21i-0640 Algorithms

Section - D

Q1

```
function areEquivalent:
        Takes three strings referenceDNA,
  equivalentRule, pairstomatch

      Set pos to the position of '-' in equivalentRule
      split equivalentRule from pos into two parts

      If length of part1 or part 2 is not equal to
      referenceDNA, return false.

      Initialize map[256] zeros.
      for each character in reference DNA:
              char from part1 → map
              char from part2 → map
      for each in pairs to match
          If index is out of bounds return false

      return true
end
```

Q2

function matchPatter, takes two strings 'text' and 'pattern'.

```
        Initialize option to 0
        ~~textlength~~ length of text → textlength.
        length of pattern → patternlength
     Loop each char in text:
        Based on option:
        If option is 0:
              If char is 'b'
                    ~~op~~ option is set to 1
                    Increment loop index
        else if option is 1:
              If char is 'a'
                    option = 2
              else
                    option = 0
        else if option is 2:
              If char is 'd'
                    return 1
              else if char is not 'a','b','c'
                    set option to 0
     end loop
end
```

Q3

```
function longestPalindrome takes one string input
        length of input -> n
      dp [50][50] set all to 0

      for len from 2 to n
          for i from 0 to (n-len)

                j is set to (len+i-1)

              If  i = j  and len is 2
                    set dp[i][j] to 2
            elseif  i = j
                    set dp[i][j] to (dp[i+1][j-1] + 2)
          else
                    set dp[i][j] to max (dp[i+1][j] and dp[i][j-1]
    end both loops
result set to empty string
set i to 0  and  j to n-1
while  i <= j
      If chars at i = j
            append char at i  to result
            i+=1 and j-=1
    else if  dp[i+1][j]  >= dp[i][j-1]
          i+=1
    else
          j-=1
    copy copy of result in half

    If i-1 >=0, j+1 < n and chars at i-1 = j+1
            remove last character from half
    append the reverse of half to result

    return result
end
```

Q4

```
function longestContigousSubstring takes one
string input
length of input -> n
maxLength set to 0
~~loo~~ for i from 0 to n
        string left set to substring of input
        string right set to substring of input
        reverse right
        Leftlength -> lenleft
        length of right -> ~~ri~~ lenRight
        dp[50][50] set to 0

        for l from 1 to ~~left~~ lenleft
            for r from lenRight
                If char at l-1 in left equals
                    char at r-1 in right
                        set dp[l][r] to (dp[l-1][r-1]+1)
                        update maxlength
                else
                        set dp[l][r] to 0
        end loop
    ~~return~~ end all loops
return max length
end
```