Wstęp do Informatyki i Programowania
Laboratorium: Lista 4

Przemysław Kobylański

Zadanie 1 (10 pkt)

Napisz funkcję match(char* wzorzec, char* łańcuch), która ustala zgodność wzorca z łańcuchem. Znak '?' we wzorcu oznacza zgodność z dowolnym innym znakiem. Znak '*' oznacza zgodność z dowolnym, również pustym, ciągiem znaków w łańcuchu. Znak różny od '?' i '*' oznacza zgodność tylko z samym sobą.

Na przykład

- match("*.doc",s) ma zwracać `true` wtedy i tylko wtedy, gdy napis s jest ciągiem dowolnych znaków z czterema ostatnimi znakami '.doc'
- match("a???",s) ma zwracać `true` wtedy i tylko wtedy, gdy s ma długość 4 i zaczyna się od litery 'a'
- match("a*b*b*c", s) ma zwracać `true` wtedy i tylko wtedy, gdy napis s zaczyna się od litery 'a' i kończy się literą 'c' a między nimi znajdują się przynajmniej dwie litery 'b' (niekoniecznie obok siebie)

Wskazówka

Pomyśl o dopasowaniu jako procesie uzgadniania kolejnych znaków:

```
match("b?ba", "baab") = match("?ba", "aab") = match("ba", "ab") =
                                                        = false

match("a??a", "abba") = match("??a", "bba") = match("?a", "ba") =
             match("a", "a") = match("", "") =  true
```

Oczywiście przypadek gwiazdki we wzorcu jest dużo ciekawszy.

Uwaga

Sprawdź działanie Twojej funkcji na następujących przykładach:

```
match("a*b*a", "abababababababababababa") = true
match("a*b*a", "abababababababababababab")  = false
```

Zadanie 2 (~~10~~ 20 pkt)

Napisz program, który gra w grę Mastermind[1].

W grze łamie się ukryty kod złożony z sekwencji czterech kolorów wybranych spośród sześciu (kolory mogą powtarzać się w sekwencji).

[1] Gra znana jest też między innymi pod nazwami SuperMind, SuperBrain.

Rysunek 1: Przykładowa partia gry.

Gra dwóch graczy. Jeden układa kod z czterech kolorów (nazywać będziemy go *koderem*) a drugi stara się go odgadnąć (nazywać będziemy go *dekoderem*).

Dekoder podaje sekwencję czterech kolorów i dostaje od kodera w odpowiedzi informację ile kolorów jest poprawnych i na swoich miejscach a ile poprawnych ale na złych miejscach.

Na rysunku 1 przedstawiono przykładową partię.

Liczbę (ale nie pozycje) kolorów poprawnych i na swoim miejscu koder oznacza w odpowiedzi kołeczkami w kolorze czerwonym, natomiast liczbę (ale nie pozycje) kolorów poprawnych ale nie na swoim miejscu kołeczkami w kolorze białym.

Celem naszym jest napisanie programu, który będzie łamał ukryty kod (grał jako dekoder).

Zamiast kolorów używać będziemy cyfr od 1 do 6.

Zakładamy, że osoba uruchamiająca program jest koderem i zapisała sobie na kartce kod złożony z czterech cyfr.

Program cyklicznie drukuje swoją propozycję kodu (cztery cyfry z zakresu od 1 do 6) i czeka na wprowadzenie przez kodera liczbę czerwonych i liczbę białych kołeczków.

Twój program nie musi łamać kodu minimalną liczbą pytań. Wystarczy, że będzie to robił w kilku pytaniach (maksymalnie ośmiu).

Nie szukaj w sieci takiego programu! Pomyśl nad własnym rozwiązaniem.

Przykłady

W poniższym przykładzie program znalazł poprawny kod po zadaniu pięciu pytać:

```
$ ./mastermind
[1] [1] [1] [1]?
  red: 1
white: 0
[1] [2] [2] [2]?
  red: 0
white: 1
[3] [1] [3] [3]?
  red: 1
white: 1
[3] [4] [1] [4]?
  red: 2
white: 2
[3] [4] [4] [1]?
  red: 4
white: 0
I win
```

W tym przykładzie program wykrył po pięciu pytaniach, że koder oszukał go:

```
$ ./mastermind
[1] [1] [1] [1]?
  red: 1
white: 0
[1] [2] [2] [2]?
  red: 1
white: 0
[1] [3] [3] [3]?
  red: 1
white: 0
[1] [4] [4] [4]?
  red: 1
white: 0
[1] [5] [5] [5]?
  red: 0
white: 1
you are cheating!
```