# რაგბი ჩვენი თამაშია (50 ქულა)

რაგბის თამაშების შედეგები გვაქვს მოცემული როგორც ბაიტების მიმდევრობა. თითოეული თამაშის შედეგი წარმოდგენილია როგორც:

<გუნდის სახელი><სფეისი><ერთ ბაიტიანი ქულა პირველი გუნდის><ერთ ბაიტიანი ქულა მეორე გუნდის><სფეისი><მეორე გუნდის სახელი>

გუნდის სახელები შედგება დიდი და პატარა ლათინური ასოებისგან. თამაშის შედეგები გამოყოფილია წერტილ-მძიმეებით. ბოლოს თამაშის შედეგი წერტილ მძიმის ნაცვლად მთავრდება ‘\0’ სიბოლოთი. მაგალითად:

ascii: “a 0 2 b;b 1 0 c”

hex: 61 20 30 20 32 20 62 3b 62 20 31 20 30 20 63 00

თქვენი ამოცანაა იმპლემენტაცია გაუკეთოთ შემდეგ სამ ფუნქციას:

// Returns total number of games played by team with given name.

int TotalGamesPlayed(void\* results, char\* team\_name);

// Returns total points scored across all games by team with given name.

int TotalPointsScored(void\* results, char\* team\_name);

// Returns number of games won by team with given name.

int NumWins(void\* results, char\* team\_name);

# კოდირება (50 ქულა)

RNE (run-length encoding) მონაცემების კომპრესირების მექანიზმია რომელიც მიყოლებით გამეორებულ ინფორმაციას ანაცვლებს მისი სიგრძით, თუ რამდენჯერ მეორდება იგი და თვითონ გამეორებადი ინფორმაციით. მაგალითად:

aabcbcbcx კომპრესირდება როგორც <1,2>a<2,3>bc<1,1>x

yfoofoobar კომპრესირდება როგორც <1,1>y<3,2>foo<3,1>bar

ვიცით რომ გამეორებადი საბსტრინგის სიგრძე და გამეორებების რაოდენობა არ აღემატება 15-ს. ეს საშუალებას გვაძლევს ორივე მონაცემი ერთ ბაიტში ჩავწეროთ რომლის პირველი 4 ბიტიც აღნიშნავს გამეორებადი ინფორმაციის სიგრძეს ხოლო ბოლო 4 ბაიტი თუ რამდენჯერ მეორდება იგი. ანუ საბოლოოდ aabcbcbcx კოდირდება როგორც:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0001***0010* | a | **0010***0011* | b | c | **0001***0001* | x |

თქვენი ამოცანაა იმპლემენტაცია გაუკეთოთ ფუნქციას რომელიც მოცემულ ‘\0’-ით დასრულებულ კომპრესირებული ინფორმაციას დეკომპრესირებას გაუკეთებს.

Decompress ფუნქციის გამოძახების შემდეგ გადაცემული char\*\* ცვლადი უნდა უთითბდეს დეკომპრესირებულ C-სტრინგს(char\*).

// Decompresses given RNE compressed data.

// Takes ownership of memory block pointed by passed data pointer.

// Char array pointed by data ends with ‘\0’.

// After returning data must be pointing to a memory block which contains decompressed

// information ownership of which is passed back to the caller.

void Decompress(char\*\* data);