Buba-kolosy

June 17, 2016

We wszystkich grupach do napisania była jedna funkcja w assemblerze która miała być wywoływana z C.

1 Czwartek 11:00

Program miał zostać napisany na platforme **i386 (32bity)** Funkcja do napisania:

```
long max_diff(long a, long b, long c, long d);
```

Która zwraca maksymalna różnice dwóch z czterech elementów. np

Zadanie sprowadza sie do znalezenia najwiekszej oraz najmniejszej liczby a nastepnie odjecia ich od siebie

2 Czwartek 12:30

Program miał zostać napisany na platforme AMD64 (64bity) Funkcja do napisania:

```
char* generate_str(char * s, int c, int n, int inc);
```

Ma generować stringa złożonego z n
 znaków c. Jeśli inc=0znaki maja być takie same, w przeciwnym przypadku znaki maja sie zwiekszać o 1. s
 jest buforem do zapisania wygenerowanego stringa, miejsce na niego powinno być zarezerwowane w programie w C.

(protip: use malloc for greater good.)

Przykład działania:

```
generate_str(char* s, 'a', 7, 0) = "aaaaaaaa"
generate_str(char* s, 'b', 3, 1) = "bcd"
```

3 Piatek 8:00

Program miał zostać napisany na platforme AMD64 (64bity) Funkcja do napisania:

```
char* fun(char* buf, char* a, char* b)
```

Ma generować stringa złożonego z sizeof(a) + sizeof(b) znaków. Ten string ma składać sie ze znaków stringa a oraz b na przemian, miejsce na wynikowego stringa powinno być zarezerwowane w programie w C. (protip: use malloc for greater good.)

Przykład działania:

```
fun(char* buf, "bb", "ua") == "buba"
fun(char* buf, "abcd", "1234") == "a1b2c3d4"
fun(char* buf, "adzjdp", "nre ua") == :)
```

4 Piatek 9:30

Program miał zostać napisany na platforme **i386 (32bity** Funkcja do napisania:

```
int max_ind(int a, int b, int c, int d)
```

Ma ona zwrócić indeks dla najwiekszego z pośród podanych argumentów. Dodatkowo, jeżeli maksymalan wartość sie powtarza, funkcja ma zwrócić 0.

Przykład działania

5 Kolos poprawkowy

Program miał zostać napisany na platforme AMD64 (64 bity) Funkcja do napisania:

```
long long max_1_ind(
long long* tab,
long long n,
long long* even_count,
long long* neg_count
)
```

Funkcja otrzymuje tablice liczb poprzez wskaźnik tab, gdzie n to liczba elementów tablicy. Dodatkowo przekazywane sa dwa wskaźniki, dla których ma być zarezerwowane miejsce na jeden element w C. Po zakończeniu działania funkcji, w pamieci wskazuwanej przez **even count**, ma znajdować sie ilość liczb parzystych wystepujacych w tablicy. W **neg count** ma sie znaleźć ilość liczb ujemnych w tablicy. I teraz wisienka na torcie. Funkcja po wykonaniu, ma zwrócić indeks elementu tablicy który posiada najwieksza liczbe jedynek w swojej reprezentacji binarnej.

Przykład użycia:

```
long long tab[7] = {-1,2,-3,4,5,-6,-7}
long long* a = malloc(sizeof(long long));
long long* b = malloc(sizeof(long long));
long long max = max_1.ind(tab, 7, a, b)
printf("%lld, %lld, %lld", a, b, max)
```

program powinien wypluć: