Wstęp do programowania w języku C (2017/2018)

Grupa MSZ

Lista 6 na zajęcia 21.11.2017

Zadanie 1 (10 pkt. na pracowni, 5 pkt. później).

Zaimplementuj funkcje:

```
void* malloc_pattern(size_t size, const void* pattern, size_t pattern_len);
void* realloc_pattern(void* t, size_t old_size, size_t new_size,
   const void* pattern, size_t pattern_len);
```

Powinny działać tak jak standardowe `malloc` i `realloc`, z tą różnicą, że nowo przydzielony obszar pamięci jest inicjalizowany cyklicznie powtarzającym się wzorcem o długości `pattern_len` bajtów podanym pod adresem `pattern`.
Przykładowy test użycia:

```
const char *p1 = "ab";
char* t = (char*)malloc_pattern(5, p1, 2);
const char *p2 = "cd";
t = (char*)realloc_pattern(t, 5, 10, p2, 2);
printf("%.10s\n", t); // ababacdcdc
```

Zadanie 2 (10 pkt.).

Zdefiniuj strukturę `Array2D`, która reprezentuje dwuwymiarową tablicę `int`'ów o modyfikowalnych rozmiarach:

```
struct Array2D {
  unsigned int width, height;
  int *t;
};
```

Wskaźnik `t` wskazuje na sekwencję `int`'ów z kolejnych wierszy.
Do obsługi tablicy zaimplementuj trzy funkcje:

```
int get_cell(const struct Array2D *array,
   unsigned int x, unsigned int y)
```

która zwraca wartość w komórce o współrzędnych `x,y`.

```
void set_cell(struct Array2D *array,
   unsigned int x, unsigned int y, int val)
```

która ustawia wartość w komórce o współrzędnych `x,y` na `val`.

```
resize(struct Array2D *array, unsigned int width, unsigned int height)
```

która ustawia nowe rozmiary tablicy w następujący sposób: komórki które mieszczą się w nowych wymiarach zachowują zawartość, pozostałe komórki są tracone; wartości nowych komórek inicjowane są zerem. W tej funkcji należy wykonać dokładnie jednego wywołanie `realloc` bez żadnych wywołań `malloc`, czyli nie wolno alokować żadnej tymczasowej pamięci; inaczej stracilibyśmy na efektywności implementacji. Zadbaj o poprawną obsługę tablic o zerowych wymiarach; w takim przypadku wskaźnik na zawartość tablicy powinien mieć wartość `NULL`.

Pustą tablicę można utworzyć przypisując odpowiednio wartości `0,0,NULL` do pól struktury.
Przykład. Załóżmy, że tablica `array` wygląda tak:

```
1,2,3
4,5,6
7,8,9
```

W tym momencie `array.width` oraz `array.height` są równe 3, a `array.t` wskazuje na sekwencję `int`'ów: 1,2,3,4,5,6,7,8,9.
Po wywołaniu `set_cell(array, 1, 0, 8)`:

```
1,8,3
4,5,6
7,8,9
```

Po wywołaniu `resize(array, 5, 2)`:

```
1,8,3,0,0
4,5,6,0,0
```

Zadanie 3 (10 pkt.). Opisane w SKOS.