Kurs C
Zestaw 10

1. Napisać funkcje służące do obsługi stosu, tzn.

- funkcję `void init (stos * s, int r)` służącą do zainicjowania stosu s przechowującego dane rozmiaru r,
- funkcję `void add (stos * s, const void * d)` dodającą na stos s dane znajdujące się pamięci wskazywanej przez d (skorzystać z funkcji `memcpy`),
- funkcję `int get (kolejka * k, void * d)` zwracającą 0, gdy w stos s jest pusta, wartość różną od w przeciwnym wypadku, oraz wpisującą do pamięci wskazywanej przez d dane pobrane ze stosu (skorzystać z funkcji `memcpy`),
- funkcję `void done (stos * s)` likwidującą stos s.

W powyższych funkcjach `stos` jest następującym typem danych

```
struct element { /* pojedynczy element */
  void * dane; /* wskaźnik do danych */
  struct element * następny;
    /* wskaźnik do następnego elementu */
  };
struct stos {
  int rozmiar; /* rozmiar danych */
  struct element * el;  /* lista elementów */
};
```

2. Napisać funkcje służące do obsługi kolejki, tzn.

- funkcję `void init (kolejka * k, int r)` służącą do zainicjowania kolejki k przechowującej dane rozmiaru r,
- funkcję `void add (kolejka * k, const void * d)` dodającą do kolejki k dane znajdujące się pamięci wskazywanej przez d (skorzystać z funkcji `memcpy`),
- funkcję `int get (kolejka * k, void * d)` zwracającą 0, gdy w kolejka k jest pusta, wartość różną od w przeciwnym wypadku, oraz wpisującą do pamięci wskazywanej przez d dane pobrane z kolejki (skorzystać z funkcji `memcpy`),

- funkcję `void done (kolejka * k)` likwidującą kolejkę k.

W powyższych funkcjach `kolejka` jest następującym typem danych

```
struct element { /* pojedynczy element */
  void * dane; /* wskaźnik do danych */
  struct element * następny;
    /* wskaźnik do następnego elementu */
  };
struct kolejka {
  int rozmiar; /* rozmiar danych */
  struct element * el;  /* lista elementów */
  struct element * l; /* ostatni element */
};
```