1. Stos, kolejka

1 Zadanie

W programie są zdefiniowane tablice `stack, queue, cbuff`. Ich rozmiary są takie same i równe 10.

1.1 Stos

Stos jest realizowany za pomocą tablicy `stack` i zmiennej `top` zdefinowanymi poza blokami funkcji. Szablon programu należy uzupełnić o definicję funkcji obsługujących stos `stack_push(), stack_pop(), stack_state()`.

1. Funkcja `stack_push(double x)` kładzie na stosie wartość parametru i zwraca stałą `OK`, a w przypadku przepełnienia stosu zwraca stałą `OVERFLOW1`.
2. Funkcja `stack_pop(void)` zdejmuje ze stosu jeden element i zwraca jego wartość. W przypadku stosu pustego zwraca stałą `NAN`.
3. Funkcja `stack_state(void)` zwraca liczbę elementów leżących na stosie, a w przypadku stosu pełnego - stałą `FULL`.
4. **Wejście**
   1 oraz ciąg liczb rzeczywistych reprezentujących operacje na stosie:
   - Liczba dodatnia powoduje dodanie jej wartości na stosie, a w przypadku przepełnienia – wypisanie tekstu OVERFLOW.
   - Liczba ujemna powoduje zdjęcie jednego elementu ze stosu i wypisanie wartości zwracanej przez funkcję `stack_pop(void)`.
   - Zero powoduje wypisanie wartości zwracanej przez funkcję `stack_state(void)` i kończy program.
5. **Wyjście**
   Ciąg wartości elementów zdejmowanych ze stosu (lub innych wartości zwracanych przez ww. funkcje) oraz w nowej linii – stan końcowy stosu.
6. **Przykład:**

```
Wejście:
1
2. 4. 5. 7. 1. -2. -1. 9. -1. 5. 0.

Wyjście:
1.00 7.00 9.00
4
```

1.2 Kolejka w tablicy z przesunięciami

Obsługa kolejki (typu FIFO) jest realizowana z zastosowaniem tablicy `queue` i zmiennej `in` zdefiniowanymi poza blokami funkcji. Wartością zmiennej `in` jest liczba klientów oczekujących w kolejce. Kolejny pojawiający się klient otrzymuje kolejny numer począwszy od 1. Klient, który zastaje pełną kolejkę, rezygnuje z

oczekiwania, ale zachowuje swój nr (kolejny klient otrzyma następny numer). Numery klientów czekających w kolejce są pamiętane w kolejnych elementach tablicy `queue` w taki sposób, że numer klienta najdłużej czekającego jest pamiętany w `queue[0]`.

Szablon programu należy uzupełnić o definicję funkcji obsługujących kolejkę `queue_push(), queue_pop(), queue_state()`.

1. Funkcja `queue_push(int in_nr)` powiększa kolejkę o `in_nr` klientów. Numer bieżącego klienta jest pamiętany w zmiennej globalnej `curr_nr`. W przypadku przepełnienia kolejki pisze słowo `OVERFLOW` – przy jednym wywołaniu funkcji – jedno słowo, niezależnie od liczby klientów, którzy zrezygnowali.
2. Funkcja `queue_pop(int out_nr)` symuluje obsługę `out_nr` najdłużej czekających klientów. W przypadku, gdy `out_nr` jest większa od długości kolejki, funkcja wypisuje słowo `UNDERFLOW` – podobnie jak funkcja `queue_push(int in_nr)` – jednokrotnie.
3. Funkcja `queue_state()` wypisuje numery czekających klientów (wg kolejności w kolejce), a w przypadku pustej kolejki pisze słowo `EMPTY`.
4. **Wejście**
   2 oraz ciąg liczb całkowitych reprezentujących operacje na kolejce:
   - Liczba dodatnia jest liczbą klientów dochodzących do kolejki.
   - Liczba ujemna jest liczbą obsłużonych klientów opuszczających kolejkę.
   - Zero powoduje wypisanie wartości zwracanej przez funkcję `queue_state(void)` i kończy program.
5. **Wyjście**
   Słowa oznaczające sytuacje "osobliwe" – `OVERFLOW UNDERFLOW` oraz ciąg liczb - numerów klientów czekających w kolejce albo słowo `EMPTY`.
6. **Przykład:**

```
Wejście:
2
1 3 5 -2 7 -3 2 0

Wyjście:
OVERFLOW 6 7 8 9 10 11 12 17 18
```

1.3 Kolejka w buforze cyklicznym

Obsługa kolejki (typu FIFO) jest realizowana z zastosowaniem tablicy `cbuff` służącej jako bufor cykliczny i zmiennych `out` i `len` zdefiniowanymi poza blokami funkcji. Wartością zmiennej `len` jest liczba klientów oczekujących w kolejce, a zmiennej `out` – indeks tablicy `cbuff`, w której jest pamiętany numer klienta najdłużej czekającego. Kolejny pojawiający się klient otrzymuje kolejny numer począwszy od 1. Klient, który zastaje pełną kolejkę, rezygnuje z oczekiwania, ale zachowuje swój nr (kolejny klient otrzyma następny numer).

Szablon programu należy uzupełnić o definicję funkcji obsługujących kolejkę `cbuff_push(), cbuff_pop(), cbuff_state()`.

1. Funkcja `cbuff_push(int cli_nr)` powiększa kolejkę o jednego klienta o numerze `cli_nr`. W przypadku przepełnienia kolejki pisze słowo `OVERFLOW`.
2. Funkcja `cbuff_pop()` symuluje obsługę najdłużej czekającego klienta. W przypadku, gdy kolejka była pusta, funkcja wypisuje słowo `UNDERFLOW`.
3. Funkcja `cbuff_state()` wypisuje numery czekających klientów (wg kolejności w kolejce), a w przypadku pustej kolejki pisze słowo `EMPTY`.

4. **Wejście**
   3 oraz ciąg liczb całkowitych reprezentujących operacje na kolejce:

   - Liczba dodatnia oznacza przyjście nowego klienta.
   - Liczba ujemna oznacza obsługę i opuszczenie kolejki przez jednego klienta.
   - Zero powoduje wypisanie wartości zwracanej przez funkcję `cbuff_state(void)` i kończy program.

5. **Wyjście**
   Słowa oznaczające sytuacje "osobliwe" – `OVERFLOW UNDERFLOW` oraz ciąg liczb - numerów klientów czekających w kolejce albo słowo `EMPTY`.

6. **Przykład:**

```
Wejście:
3
1 3 5 -2 7 -3 2 0

Wyjście:
3 4 5
```