2. 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(int n)` kładzie na stosie wartość parametru i zwraca stałą `OK`, a w przypadku przepełnienia nie zmienia zawartości stosu i zwraca stałą `OVERFLOW`.
2. Funkcja `stack_pop(void)` zdejmuje ze stosu jeden element i zwraca jego wartość. W przypadku stosu pustego zwraca stałą `UNDERFLOW`.
3. Funkcja `stack_state(void)` zwraca liczbę elementów leżących na stosie.

Wejście
1 oraz ciąg liczb całkowitych reprezentujących operacje na stosie:

- Liczba dodatnia `n` powoduje dodanie jej wartości na stos (wywołanie `stack_push(n)`).
- Liczba ujemna powoduje zdjęcie jednego elementu ze stosu (wywołanie `stack_pop(void)`).
- Zero powoduje wywołanie funkcji `stack_state(void)` i zakończenie programu.

Wyjście
Ciąg wartości elementów zdejmowanych ze stosu (lub innych wartości zwracanych przez ww. funkcje) oraz stan końcowy stosu.

Przykład:

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

Wyjście:
```
1 7 9
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 numer (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(), queue_print()`.

1. Funkcja `queue_push(int in_nr)` powiększa kolejkę o `in_nr` klientów. Numer bieżącego klienta jest pamiętany w zmiennej globalnej `curr_nr`. Zwraca wartość stałej `OK` w przypadku poprawnego dopisania wszystkich klientów do kolejki. W przypadku, gdy liczba wchodzących do kolejki jest większa niż liczba wolnych miejsc w kolejce, miejsca w kolejce są zajmowane do zapełnienia miejsc, a pozostali "niedoszli klienci" rezygnują (zachowując swoje numery). W takiej sytuacji funkcja zwraca stałą `OVERFLOW`.
2. Funkcja `queue_pop(int out_nr)` symuluje obsługę `out_nr` najdłużej czekających klientów. Funkcja zwraca długość pozostałej kolejki. W przypadku, gdy `out_nr` jest większa od długości kolejki, funkcja zwraca stałą `UNDERFLOW`.
3. Funkcja `queue_state()` liczbę klientów czekających w kolejce.
4. Funkcja `queue_print()` wypisuje numery czekających klientów (w kolejności wejścia do kolejki).

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 wywołanie funkcji `queue_state(void)` i wypisanie zwracanej przez nią wartości, wywołanie funkcji `queue_print()` oraz zakończenie testu.

Wyjście
Wartości oznaczające sytuacje "osobliwe" – `OVERFLOW UNDERFLOW`

Po wpisaniu na wejściu liczby 0 są wypisywane: liczba czekających klientów oraz ich numery wg kolejności w kolejce.

Przykład:

Wejście:
```
2
1 3 5 -2 7 -3 2 0
```

Wyjście:
```
-2
9
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, który jest zapisywany do elementu tablicy (bufora) o indeksie `out + len` (z uwzględnieniem "cykliczności" bufora). Klient, który zastaje pełną kolejkę, rezygnuje z oczekiwania, ale zachowuje swój numer (kolejny klient otrzyma następny numer).

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

1. Funkcja `cbuff_push(int cli_nr)` powiększa kolejkę o jednego klienta o numerze `cli_nr` i zwraca stałą `OK`. W przypadku przepełnienia kolejki zwraca wartość `OVERFLOW`.
2. Funkcja `cbuff_pop()` symuluje obsługę najdłużej czekającego klienta. Funkcja zwraca numer klienta wychodzącego z kolejki, a w przypadku, gdy kolejka była pusta, zwraca stałą `UNDERFLOW`.
3. Funkcja `cbuff_state()` zwraca liczbę czekających klientów
4. Funkcja `cbuff_print()` wypisuje numery czekających klientów (wg kolejności w kolejce).

Wejście
3 oraz ciąg liczb całkowitych reprezentujących operacje na kolejce:

- Liczba dodatnia oznacza przyjście nowego klienta. Pierwszy klient otrzymuje numer 1. Kolejny klient otrzymuje kolejny numer. Klient, który zastaje pełną kolejkę, rezygnuje z oczekiwania, ale zachowuje swój numer (kolejny klient otrzyma następny numer). Klient (jego numer) jest umieszczany w kolejce przez wywołanie funkcji `cbuff_push(cli_nr)`. Funkcja ta zapisuje przesłany numer w elemencie tablicy (bufora) o indeksie `out + len` (z uwzględnieniem "cykliczności" bufora).
- Liczba ujemna oznacza obsługę i opuszczenie kolejki przez jednego klienta (wywołanie funkcji `cbuff_pop()`).
- Zero powoduje wywołanie funkcji `cbuff_state()` i wypisanie zwracanej przez nią wartości, wywołanie funkcji `cbuff_print()` oraz zakończenie testu.

Wyjście
Pierwsza linia standardowego wyjścia zawiera numery klientów wychodzących z kolejki oraz stałe `OVERFLOW` lub `UNDERFLOW` w kolejności wg czasu zdarzeń. W drugiej linii wypisywana jest liczba klientów pozostających w kolejce po zakończeniu symulacji, a w trzeciej linii – ich numery.

Przykład:

Wejście:
```
3
1 1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1 0
```

Wyjście:
```
1 2 -1 -2 -2 3
10
4 5 6 7 8 9 10 11 12 15
```