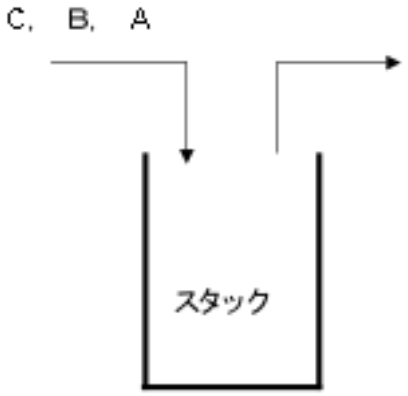
＜演習課題＞　スタック

【課題問題１】

下図に示すように、スタックに A,B,C の順序で入力されるデータがある。各データについてスタックへのプッシュとポップを一度ずつ任意のタイミングで可能とする場合、データの出力順序が何通りあるか、その数とそれぞれの出力順序(を答えよ。



解答欄　　データの出力順序　　＿＿＿＿通り

出力順序　(1)　C, B, A
(2)
(3)
(4)
(5)
(6)

【課題問題２】

中置記法で記述された(1)(2)(3)式を後置記法(逆ポーランド記法)で表しなさい

(1) (A+B)×C-D　　→

(2) (A-B)÷(C+D)×E　　→

(3) (A+B)×(C-D)+E-F×G　　→

後置記法で記述された(4)(5)(6)式を中置記法に変換して計算しなさい

(4)　6　2　÷　4　×　8　+　　→

(5)　3　5　+　5　3　−　÷　3　+　　→

(6)　9　7　3　+　×　3　5　×　÷　5　+　　→

【課題問題３】

スタックとキューの二つのデータ構造がある。(1)および(2)の手続きを順に実行した場合、変数 x へ代入されるデータはどれか。また、手続き終了時のスタックおよびキューの状態を記入すること。

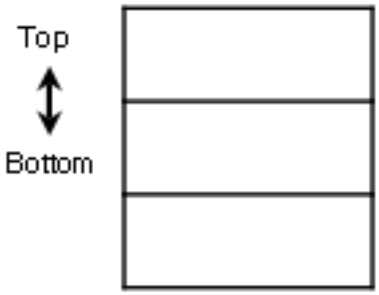
スタックとキューのサイズはどちらも 3 とし、スタックは Bottom 側から、キューは Front 側からデータを格納するものとする。また、キューはリングバッファで実現されている。

・データ a をスタックに挿入することを push(a)
・スタックからデータを取り出すことを pop()
・データ b をキューに挿入することを enq(b)
・キューからデータを取り出すことを deq()

と表すものとする。

(1) push(A)→push(B) →enq(pop())→ enq(C) →push(D) →push(deq())→ x=pop()

x=

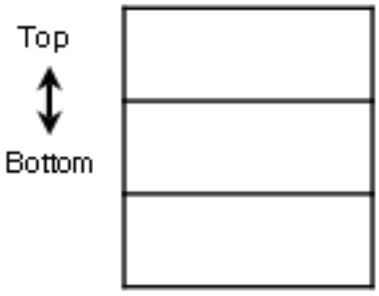


スタック

キュー

(2) enq(A)→enq(B) →push(C)→ push(D) →push(deq())→enq(pop())→push(deq())→enq(E)→pop()
→enq(F)→ x=deq()

x=



スタック

キュー

【課題問題４】

スタックを実現するための配列 stack[ ]をグローバル変数として宣言したものを右に示す。ここで、スタックの最大サイズは 5 とし、スタックポインタは-1 に初期化している。

```
/*  スタックを実現する配列 */
#define STACK_SIZE 5
#define NO_DATA -1

int stack[STACK_SIZE];
int sp=-1;
```

以下の各関数を作成しなさい。

1）スタックをプッシュする関数：int Push(int data)
　引数の整数データをスタックに積み上げ、スタックポインタをインクリメントする。プッシュが成功した場合は 0 を返し、スタックがオーバーフローした場合は-1 を返す。

2）スタックをポップする関数：int Pop( )
　スタックからデータを取り出し、スタックポインタをデクリメントする。ポップが成功した場合は取り出した整数を返し、スタックにデータがない場合は-1 を返す。

3）スタックを底から頂上へ向けて表示する関数：void ShowStack( )
　スタックに積み上がったデータを順に表示する。

上記の各関数を使用して、スタックにデータをプッシュおよびポップするメインプログラムを作成し、以下のスタック動作を行いなさい。

1）スタックを NO_DATA で初期化する。
2）スタックにデータ((例えば 111,222,333,444,555)を順にプッシュする。
3）適当にポップおよびプッシュを行うことにより、スタックの機能を確認する。
4）プッシュおよびポップを行った際にスタックにある全データを表示する。

```
u256000@imac-000[160]: ./a.out
(1) Push  (2) Pop  (0) Exit :1
Data:111
Stack : [111]
(1) Push  (2) Pop  (0) Exit :1
Data:222
Stack : [111][222]
(1) Push  (2) Pop  (0) Exit :1
Data:333
Stack : [111][222][333]
(1) Push  (2) Pop  (0) Exit :1
Data:444
Stack : [111][222][333][444]
(1) Push  (2) Pop  (0) Exit :1
Data:555
Stack : [111][222][333][444][555]
(1) Push  (2) Pop  (0) Exit :1
Data:666
Stack Overflow!
Stack Push is failed!
(1) Push  (2) Pop  (0) Exit :2
Pop data is 555
Stack : [111][222][333][444]
(1) Push  (2) Pop  (0) Exit :2
Pop data is 444
Stack : [111][222][333]
```

```
(1) Push  (2) Pop  (0) Exit :1
Data:666
Stack : [111][222][333][666]
(1) Push  (2) Pop  (0) Exit :2
Pop data is 666
Stack : [111][222][333]
(1) Push  (2) Pop  (0) Exit :2
Pop data is 333
Stack : [111][222]
(1) Push  (2) Pop  (0) Exit :2
Pop data is 222
Stack : [111]
(1) Push  (2) Pop  (0) Exit :2
Pop data is 111
Stack :
(1) Push  (2) Pop  (0) Exit :2
Stack Empty!
Stack Pop is failed!
(1) Push  (2) Pop  (0) Exit :0
u256000@imac-000[161]:
```

【課題問題５】

リングバッファを使用してキューを実現する配列 queue[ ]をグローバル変数として宣言する例を右に示す。ここで，リングバッファのサイズは 5 とし、キューのフロントカーソル(front)とリアカーソル(rear)は-1 に、キューに入っているデータ数(num)は 0 に初期化する。

```
/*  キューを実現する配列*/
#define QUEUE_SIZE 5
#define NO_DATA -1

int queue[QUEUE_SIZE];
int front=-1;
int rear=-1;
int num=0;
```

以下の各関数を作成しなさい。

１）キューをエンキューする関数: int EnQueue(int data)
　引数の整数データをキューの入れ、リアカーソルをインクリメントする。エンキューが成功した場合は 0 を返し、キューがオーバーフローした場合は-1 を返す。フロントとリアのカーソルはリングバッファに対応させる。

２）キューをデキューする関数: int DeQueue( )
　キューからデータを取り出し、フロントカーソルをインクリメントする。デキューが成功した場合は取り出した整数を返し、キューにデータがない場合は-1 を返す。

３）キューに入っているデータを表示する関数: void ShowQueue( )
　キューに入っているデータを配列の添え字順に表示する。なお、データが入っていない配列も表示する。

上記の各関数を使用して、キューにデータをエンキューおよびデキューするメインプログラムを作成し、以下のキュー動作を行いなさい。また、実行例に示すようにリングバッファの機能を確認しなさい。

１）キューを NO_DATA で初期化する。
２）キューにデータ(例えば 111,222,333,444,555)を順にエンキューする。
３）適当にデキューおよびエンキューを行い、キューのリングバッファ機能を確認する。
４）エンキュー及びデキューを行った際にキューにあるデータを表示する。

```
u256000@imac-000[164]: ./a.out
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:111
Queue : [111][   ][   ][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:222
Queue : [111][222][   ][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:333
Queue : [111][222][333][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:444
Queue : [111][222][333][444][   ]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:555
Queue : [111][222][333][444][555]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:666
Queue is Full!
Enqueue is failed!
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 111
Queue : [   ][222][333][444][555]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 222
```

```
Queue : [   ][   ][333][444][555]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 333
Queue : [   ][   ][   ][444][555]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:666
Queue : [666][   ][   ][444][555]
(1) Enqueue (2) Dequeue (0) Exit :1
 Data:777
Queue : [666][777][   ][444][555]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 444
Queue : [666][777][   ][   ][555]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 555
Queue : [666][777][   ][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 666
Queue : [   ][777][   ][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :2
Dequeue data is 777
Queue : [   ][   ][   ][   ][   ]
(1) Enqueue (2) Dequeue (0) Exit :2
Queue is Empty!
Dequeue is failed!
(1) Enqueue (2) Dequeue (0) Exit :0
u256000@imac-000[165]:
```

【課題問題６】

スタックおよびキュー両方のデータ構造を取り扱うことができるプログラムを作成せよ。具体的には課題問題 2 に示すような､スタックとキューのデータやりとり(ポップしたスタックのデータをエンキューする、およびデキューしたキューのデータをプッシュする)を可能にすること。

スタックおよびキューのサイズはどちらも 5 とし、キューはリングバッファで実現するものとする。プログラムの実行例を以下に示す。各自テスト設計を行い、プログラムが正常に動いているかどうかを確認すること。

```
u256000@imac-000[168]: ./a.out
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :1
Data:111
Stack : [111]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :1
Data:222
Stack : [111][222]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :3
 Data:333
Queue : [333][   ][   ][   ][   ]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :3
 Data:444
Queue : [333][444][   ][   ][   ]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :1
Data:555
Stack : [111][222][555]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :5
Dequeue data is 333
Queue : [   ][444][   ][   ][   ]
Stack : [111][222][555][333]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :6
Pop data is 333
Stack : [111][222][555]
Queue : [   ][444][333][   ][   ]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :3
 Data:666
Queue : [   ][444][333][666][   ]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :5
Dequeue data is 444
Queue : [   ][   ][333][666][   ]
Stack : [111][222][555][444]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :5
Dequeue data is 333
Queue : [   ][   ][   ][666][   ]
Stack : [111][222][555][444][333]
(1) Push  (2) Pop  (3) Enq (4) Deq (5) Push(Deq)  (6) Enq(Pop)  (0) Exit :0
u256000@imac-000[169]:
```