1. Create Cycle

想法:

首先，先創建 node 及他的指向，然後創建一個 malloc，再創建資料，之後使用迴圈將資料的的尾巴指向下一個資料，迴圈裡面用當 i 等於最後一個時，最後一個資料的尾巴把他的下一個資料指回到迴圈一開始的地方，

最終版 code:

```
#include <stdio.h>
#include <stdlib.h>


struct Data {
	int data1;
	int data2;
	int data3;
	int data4;
	int data5;
	int data6;
};
typedef struct Data D;


struct node {
	int data;
	struct node* nextNodeAddr;
	struct node *next;// store an address that stores struct node
};
typedef struct node N;


struct List {
	N* head; // the address of the first node of the list
	N* tail; // the address of the last node of the list
};
typedef struct List L;
```

```
N* CreateNode(int data, N* nextNodeAddr);
N* AppendData(N* prevNodeAddr, int newData);

L CreateEmptyList() {
	L l;
	l.head = NULL;
	l.tail = NULL;

	return l;
}


N* nextNodeAddr(N* nextNodeAddr) {
	N* n = malloc(sizeof(N));
	if (n == NULL) {
		return NULL;
	}

	n->nextNodeAddr = nextNodeAddr;

	return n;
}


void PrintList(N* firstNodeAddr) {
	for (N* nodeAddr = firstNodeAddr;
		nodeAddr != NULL;
		nodeAddr = (*nodeAddr).nextNodeAddr) {
		printf("%d -> ", (*nodeAddr).data);
	}
}

int main(){
	struct node node0={0,NULL};
	struct node node1={1,NULL};
	struct node node2={2,NULL};
	struct node node3={3,NULL};
```

```
        struct node node4={4,NULL};
        struct node node5={5,NULL};
        struct node node6={6,NULL};
        node0.nextNodeAddr = &node1;
        N * firstNodeAddr= node0.nextNodeAddr;
        node1.nextNodeAddr = &node2;
        node2.nextNodeAddr = &node3;
        node3.nextNodeAddr = &node4;
        node4.nextNodeAddr = &node5;
        node5.nextNodeAddr = &node6;
        node6.nextNodeAddr = &node3;
        struct node *temp = NULL;
        temp = firstNodeAddr;
        for(;temp != NULL;){
        PrintList(temp);
        temp = temp ->nextNodeAddr;
        }
}
```

2. Cycle Detection

想法:

創造兩個 node，一個一次跳兩個，另一個一次跳一個，然後在 int main 套入別人寫的 Cycle 當作測是，之後寫寫 if else，當兩個 Node 數字一樣，則停止並輸出 1，否則則繼續，最後寫 while 若當指向 null 時，一樣停止並輸出 0。

最終版 Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
	int data;
	struct node* nextNodeAddr;
};
typedef struct node N;

struct node *head=NULL;

struct node* HasCycle()
{
		struct node *pslow=head;
		struct node *pfast=head;
		while(pfast!=NULL)
		{
			pfast=pfast->nextNodeAddr;
			if(pfast!=NULL)
			{
				pfast=pfast->nextNodeAddr->nextNodeAddr;
				pslow=pslow->nextNodeAddr;
			}

			if(pfast== pslow)
			{
				printf("1 : it contain cycle\n");
			}
		}
		printf("0 : it doesn't contain cycle\n");
}
```

```
int main()
{
        struct node *node1 = (struct node*)malloc(sizeof(struct node));
        struct node *node2 = (struct node*)malloc(sizeof(struct node));
        struct node *node3 = (struct node*)malloc(sizeof(struct node));
        struct node *node4 = (struct node*)malloc(sizeof(struct node));

        head=node1;
        node1->data=1;
        node1->nextNodeAddr=node2;

        node2->data=2;
        node2->nextNodeAddr=node3;

        node3->data=3;
        node3->nextNodeAddr=node4;

        node4->data=4;
        node4->nextNodeAddr=node1;

        HasCycle();

        return 0;
}
```