자료구조 보고서

Homework#8

학과 : 소프트웨어학과
학번 : 2016039040
이름 : 윤용진

1. doubly circular linked list 구현에 관한 것이다.

(a) circular-linked-list.c의 다음 함수를 완성한다.

```
int freeList(listNode* h);
int insertLast(listNode* h, int key);
int deleteLast(listNode* h);
int insertFirst(listNode* h, int key);
int deleteFirst(listNode* h);
int invertList(listNode* h);
int insertNode(listNode* h, int key);
int deleteNode(listNode* h, int key);
```

(b) 이해한 부분을 주석으로 남긴다.

```
/*
 *  doubly circular linked list
 *
 *  Data Structures
 *
 *  Department of Computer Science
 *  at Chungbuk National University
 *
 */

#include<stdio.h>
#include<stdlib.h>
/* 필요한 헤더파일 추가 */

typedef struct Node {
        int key;
        struct Node* llink;
        struct Node* rlink;
} listNode;

/* 함수 리스트 */
int initialize(listNode** h);
int freeList(listNode* h);
int insertLast(listNode* h, int key);
int deleteLast(listNode* h);
int insertFirst(listNode* h, int key);
int deleteFirst(listNode* h);
int invertList(listNode* h);

int insertNode(listNode* h, int key);
int deleteNode(listNode* h, int key);
```

```
void printList(listNode* h);

int main()
{
        char command;
        int key;
        listNode* headnode=NULL;

        printf("[----- [Yoon YongJin]  [2016039040] -----]\n");

        do{

printf("----------------------------------------------------------------\n");
                printf("                              Doubly  Circular  Linked  List
 \n");

printf("----------------------------------------------------------------\n");
                printf(" Initialize     = z            Print          = p \n");
                printf(" Insert Node    = i            Delete Node    = d \n");
                printf(" Insert Last    = n            Delete Last    = e\n");
                printf(" Insert First   = f            Delete First   = t\n");
                printf(" Invert List    = r            Quit           = q\n");

printf("----------------------------------------------------------------\n");


                printf("Command = ");
                scanf(" %c", &command);

                switch(command) {
                case 'z': case 'Z':
                        initialize(&headnode);
                        break;
                case 'p': case 'P':
                        printList(headnode);
                        break;
                case 'i': case 'I':
                        printf("Your Key = ");
                        scanf("%d", &key);
                        insertNode(headnode, key);
                        break;
                case 'd': case 'D':
                        printf("Your Key = ");
```

```
					scanf("%d", &key);
					deleteNode(headnode, key);
					break;
				case 'n': case 'N':
					printf("Your Key = ");
					scanf("%d", &key);
					insertLast(headnode, key);
					break;
				case 'e': case 'E':
					deleteLast(headnode);
					break;
				case 'f': case 'F':
					printf("Your Key = ");
					scanf("%d", &key);
					insertFirst(headnode, key);
					break;
				case 't': case 'T':
					deleteFirst(headnode);
					break;
				case 'r': case 'R':
					invertList(headnode);
					break;
				case 'q': case 'Q':
					freeList(headnode);
					break;
				default:
					printf("\n       >>>>>   Concentration!!   <<<<<     \n");
					break;
				}

		}while(command != 'q' && command != 'Q');

		return 1;
}

int initialize(listNode** h) {

		/* headNode가 NULL이 아니면, freeNode를 호출하여 할당된 메모리 모두 해제 */
		if(*h != NULL)
				freeList(*h);

		/* headNode에 대한 메모리를 할당하여 리턴 */
		*h = (listNode*)malloc(sizeof(listNode));
```

```c
	(*h)->rlink = *h;
	(*h)->llink = *h;
	(*h)->key = -9999;
	return 1;
}

/* 메모리 해제 */
int freeList(listNode* h){
	listNode* p;
	listNode* prev;

	p = h->rlink;

	while (p != NULL && p != h)
	{
		//if (p->llink != h) prev = p->llink;
		//else
		/* 현재 탐색중인 노드를 기억 */
		prev = p;
		p = p->rlink;

		if (prev != h) free(prev);
	}
	free(h);

	return 0;
}

void printList(listNode* h) {
	int i = 0;
	listNode* p;

	printf("\n---PRINT\n");

	if(h == NULL) {
		printf("Nothing to print....\n");
		return;
	}

	p = h->rlink;

	while(p != NULL && p != h) {
		printf("[ [%d]=%d ] ", i, p->key);
		p = p->rlink;
		i++;
	}
	printf("  items = %d\n", i);


	/* print addresses */
	printf("\n---checking addresses of links\n");
	printf("-------------------------------\n");
	printf("head node: [llink]=%p, [head]=%p, [rlink]=%p\n", h->llink, h, h->rlink);

	i = 0;
	p = h->rlink;
	while(p != NULL && p != h) {
		printf("[ [%d]=%d ] [llink]=%p, [node]=%p, [rlink]=%p\n", i, p->key,
p->llink, p, p->rlink);
		p = p->rlink;
		i++;
	}

}

/**
 * list에 key에 대한 노드하나를 추가
 */
int insertLast(listNode* h, int key) {
	/* 전처리 */
	if (h == NULL) {
		printf("Initialize First\n");
		return 0;
	}

	listNode* p;
	listNode* lastnode = (listNode*)malloc(sizeof(listNode));

	/* 리스트가 비어있고, 첫 번째 노드 생성일 경우 */
	if (h->rlink == h)
	{
		insertFirst(h, key);
		return 0;
	}

	p = h->rlink;
```

```
        while (p != NULL && p != h) {
                /* 마지막 노드를 탐색 */
                if (p->rlink == h)
                {
                        lastnode->key = key;
                        /* 새로운 노드를 리스트 끝에 삽입 */
                        lastnode->llink = p;
                        lastnode->rlink = h;
                        p->rlink = lastnode;
                        h->llink = lastnode;

                        break;
                }
                p = p->rlink;
        }

        return 0;
}

/**
 * list의 마지막 노드 삭제
 */
int deleteLast(listNode* h) {
        /* 전처리 */
        if (h == NULL) {
                printf("Initialize First\n");
                return 0;
        }

        if (h->rlink == h)
        {
                printf("There is no Node to Delete\n");
                return 0;
        }

        listNode* p;
        listNode* prev;

        p = h->rlink;

        while (p != NULL && p!=h) {
                /* 리스트에 노드가 하나뿐인 경우 */
                if (p->llink == h && p->rlink == h)
                {
                        deleteFirst(h);
                        return 0;
                }
                /* 마지막 노드 삭제 */
                else if (p->rlink == h)
                {
                        prev = p->llink;
                        prev->rlink = h;
                        h->llink = prev;
                        free(p);
                        return 0;
                }
                p = p->rlink;
        }

        return 0;
}

/**
 * list 처음에 key에 대한 노드하나를 추가
 */
int insertFirst(listNode* h, int key) {
        /* 전처리 */
        if (h == NULL) {
                printf("Initialize First\n");
                return 0;
        }

        listNode* firstnode = (listNode*)malloc(sizeof(listNode));

        firstnode->key = key;
        firstnode->llink = h;
        firstnode->rlink = h->rlink;

        /*
        *  h->rlink->llink=firstnode; 는 첫 노드 생성시에는 h->llink=firstnode로 작동
        *  이후에는 기존 첫 노드의 왼쪽에 새로운 노드 삽입으로 작동
        */
        h->rlink->llink = firstnode;
        h->rlink = firstnode;
```

```c
        return 0;
}

/**
 * list의 첫번째 노드 삭제
 */
int deleteFirst(listNode* h) {
        /* 전처리 */
        if (h == NULL) {
                printf("Initialize First\n");
                return 0;
        }

        if (h->rlink == h)
        {
                printf("There is no Node to Delete\n");
                return 0;
        }

        listNode* p;

        p = h->rlink;

        /* 노드가 하나뿐인 리스트의 경우 */
        if (p->rlink == h)
        {
                h->rlink = h;
                h->llink = h;
        }
        /* 첫번째 노드 삭제 */
        else
        {
                listNode* secondnode;
                secondnode = p->rlink;
                secondnode->llink = h;
                h->rlink = secondnode;
        }

        free(p);
        return 0;
}

/**
```

```
 * 리스트의 링크를 역순으로 재 배치
 */
int invertList(listNode* h) {
	/* 전처리 */
	if (h == NULL) {
		printf("Initialize First\n");
		return 0;
	}

	if (h->rlink == h)
	{
		printf("Make List First\n");
		return 0;
	}


	listNode* nextnode = NULL;

	listNode* lastnode = NULL;

	listNode* p;

	p = h->rlink;

	/* doubly-linked-list와 동일한 처리 */
	while (p != NULL && p != h)
	{
		if (p->rlink == h) lastnode = p;

		nextnode = p->rlink;
		p->rlink = p->llink;
		p->llink = nextnode;

		p = nextnode;
	}

	h->rlink = lastnode;

	return 0;
}

/**
 *  리스트를 검색하여, 입력받은 key보다 큰값이 나오는 노드 바로 앞에 삽입
 **/
int insertNode(listNode* h, int key) {
	/* 전처리 */
	if (h == NULL) {
		printf("Initialize First\n");
		return 0;
	}

	listNode* p;
	listNode* node2insert = (listNode*)malloc(sizeof(listNode));

	/* 리스트가 비어있고, 첫 번째 노드 생성일 경우 */
	if (h->rlink == h)
	{
		insertFirst(h, key);
		return 0;
	}

	p = h->rlink;

	while (p != NULL && p != h) {
		/* 새로운 노드의 삽입 위치가 리스트의 첫 노드 앞인 경우 */
		if (p == h->rlink && p->key > key)
		{
			insertFirst(h, key);
			return 0;
		}
		/* 현재 탐색중인 노드의 키값이 입력받은 키값보다 큰 경우 */
		else if (p->key > key)
		{
			node2insert->key = key;

			node2insert->llink = p->llink;
			p->llink->rlink = node2insert;

			node2insert->rlink = p;
			p->llink = node2insert;

			return 0;
		}
		p = p->rlink;
	}
	/* 입력받은 키 값보다 큰 키 값을 갖는 노드가 없는 경우 마지막 노드로 새 노드 추
가 */
	insertLast(h, key);

	return 0;
}

/**
 * list에서 key에 대한 노드 삭제
 */
int deleteNode(listNode* h, int key) {
	/* 전처리 */
	if (h == NULL) {
		printf("Initialize First\n");
		return 0;
	}

	if (h->rlink == h)
	{
		printf("There is no Node to Delete\n");
		return 0;
	}

	listNode* p;

	p = h->rlink;

	while (p != NULL && p != h) {
		/* 삭제할 노드가 첫 번째 노드인 경우 */
		if (p == h->rlink && p->key == key)
		{
			deleteFirst(h);
			return 0;
		}
		/* 삭제할 노드가 마지막 노드인 경우 */
		else if (p->rlink == h && p->key == key)
		{
			deleteLast(h);
			return 0;
		}
		/* 리스트에서 처음으로 마주치는 키 값이 일치하는 노드를 삭제 */
		else if (p->key == key)
		{
```

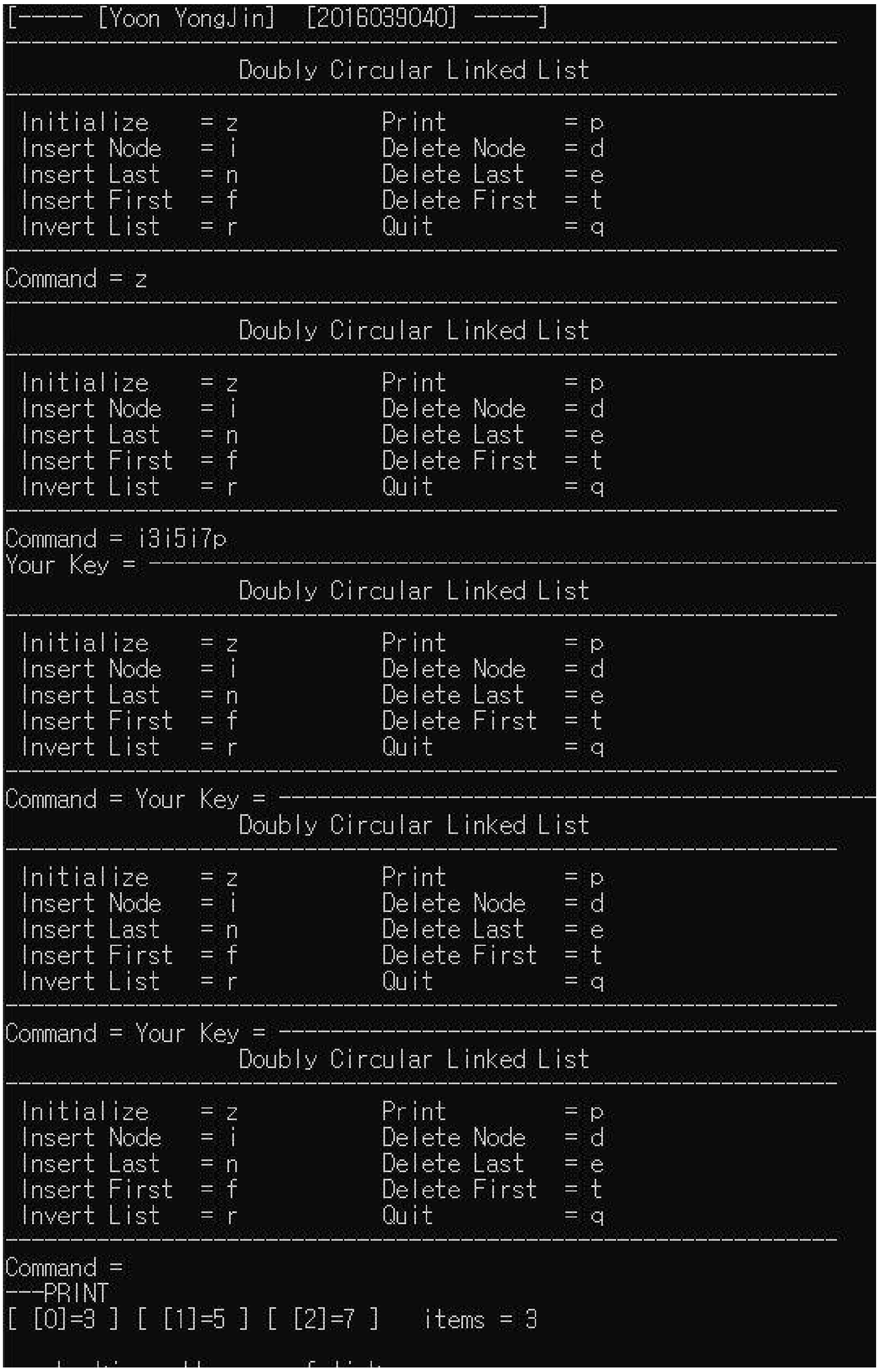
```
                        p->llink->rlink = p->rlink;
                        p->rlink->llink = p->llink;
                        free(p);
                        return 0;
                }
                p = p->rlink;
        }
        /* 모든 노드 탐색이 끝날 때까지 삭제할 노드를 찾지 못했다면 */
        printf("There is no node corresponding to the key value.\n");

        return 0;
}
```

4. GitHub에 hw7 Repository를 생성하고 doubly-linked-list.c를 업로드 한다.

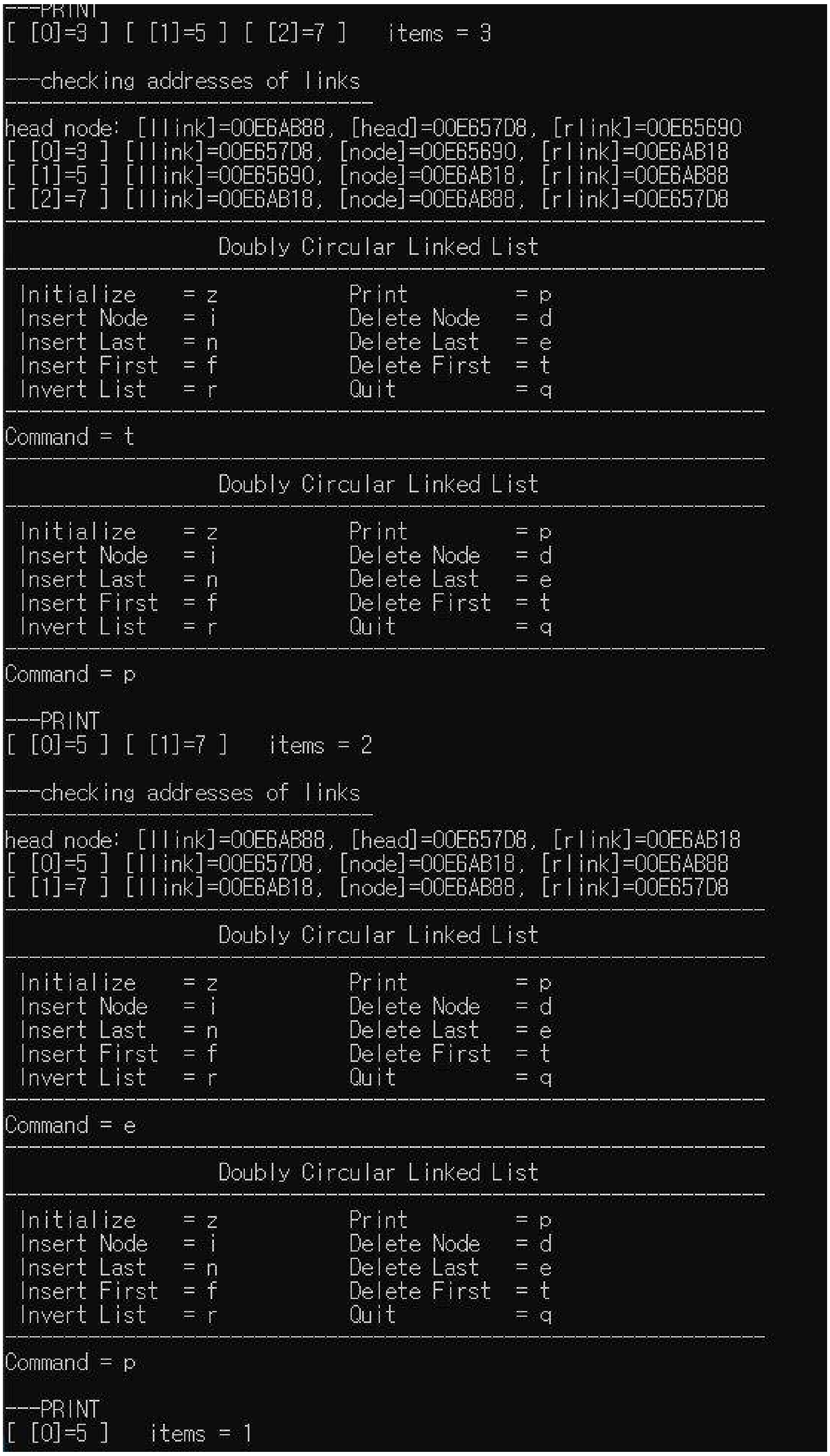
https://github.com/uniz21/DataStructure-HW-8

7. 보고서에 실행결과를 Screen Capture하여 첨부한다.

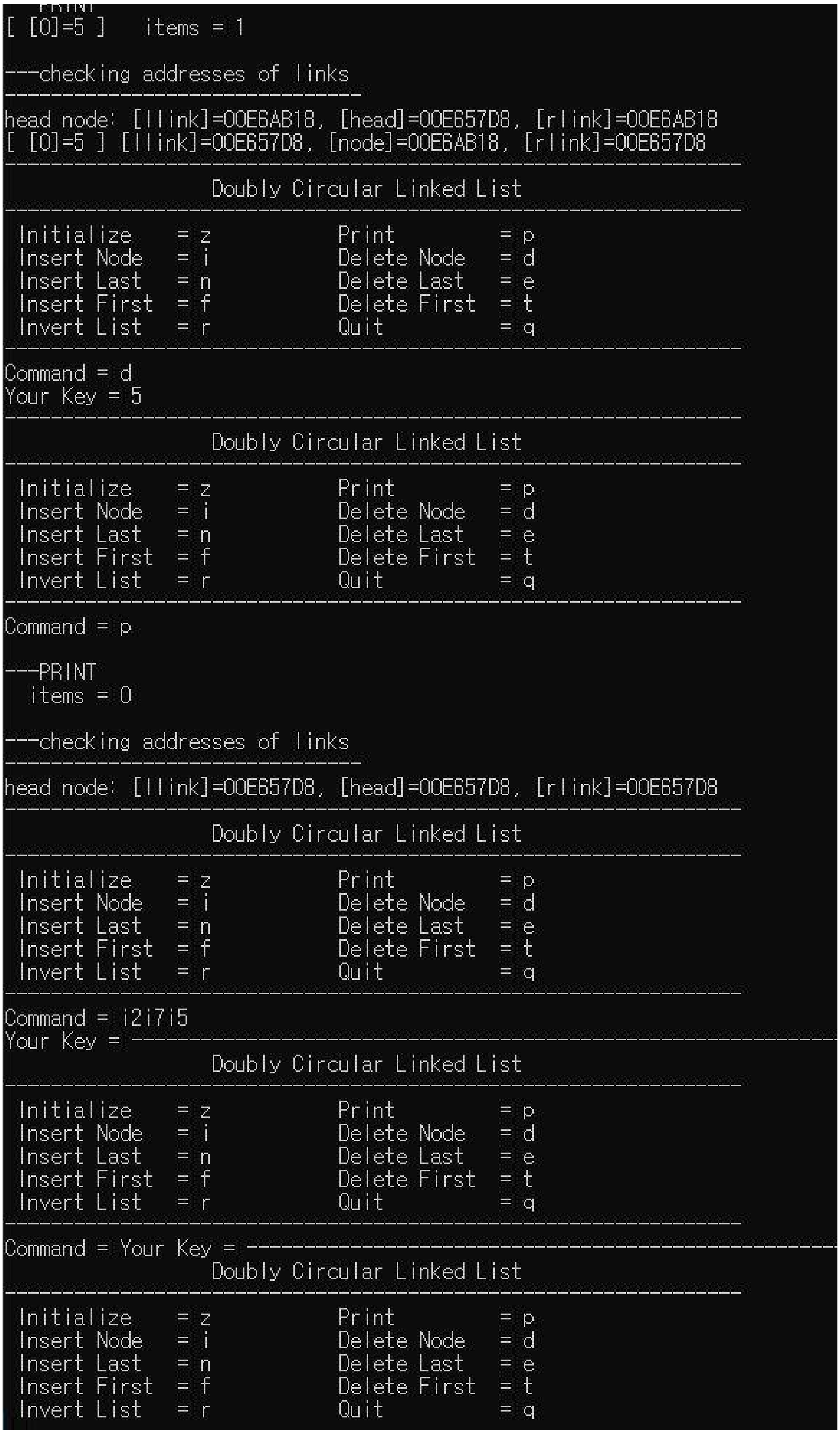
실행결과



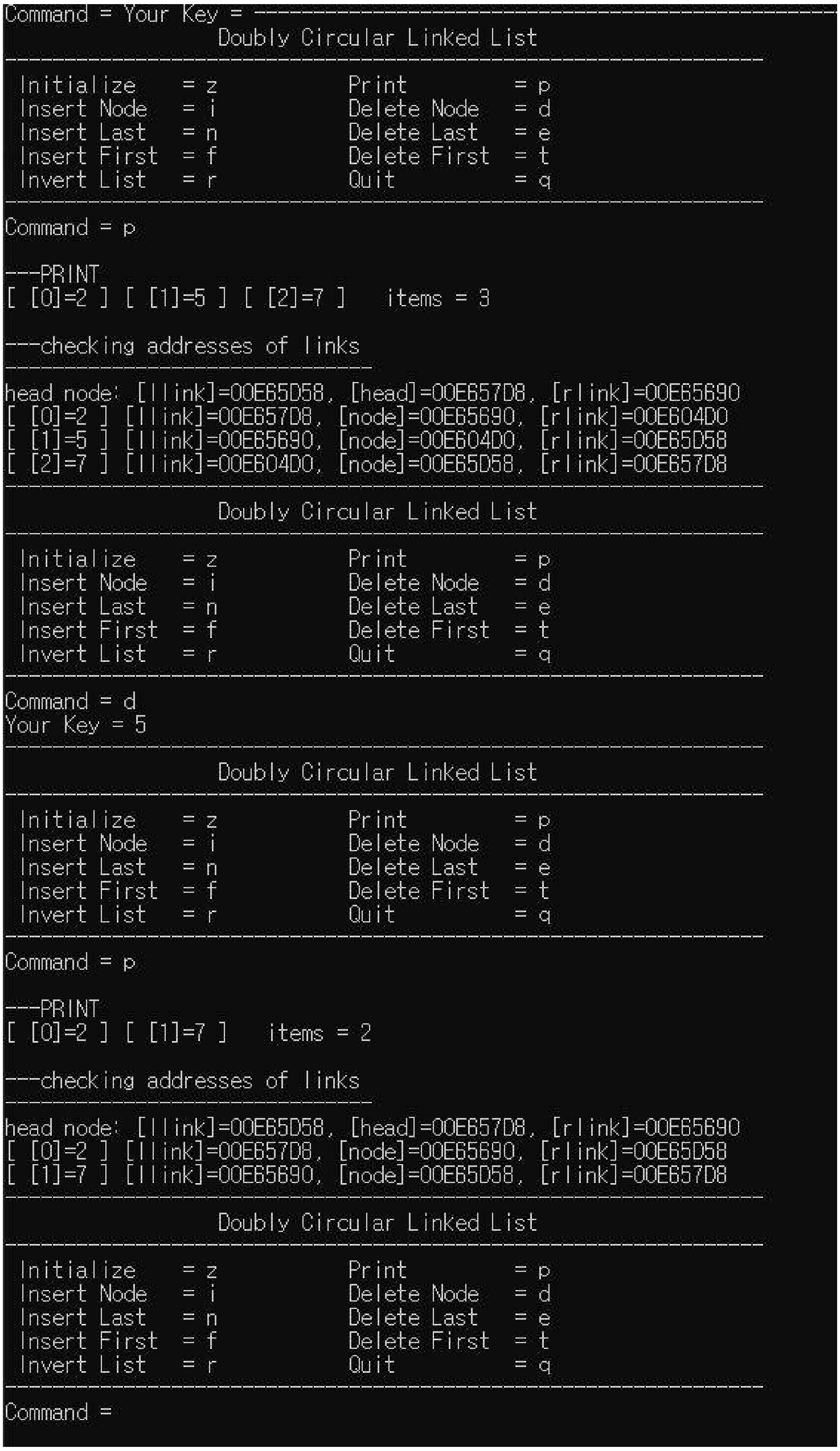
```
[----- [Yoon YongJin]  [2016039040] -----]
-----------------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------
Command = z
-----------------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------
Command = i3i5i7p
Your Key = -------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------
Command = Your Key = ---------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------
Command = Your Key = ---------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------
Command =
---PRINT
[ [0]=3 ] [ [1]=5 ] [ [2]=7 ]   items = 3
```



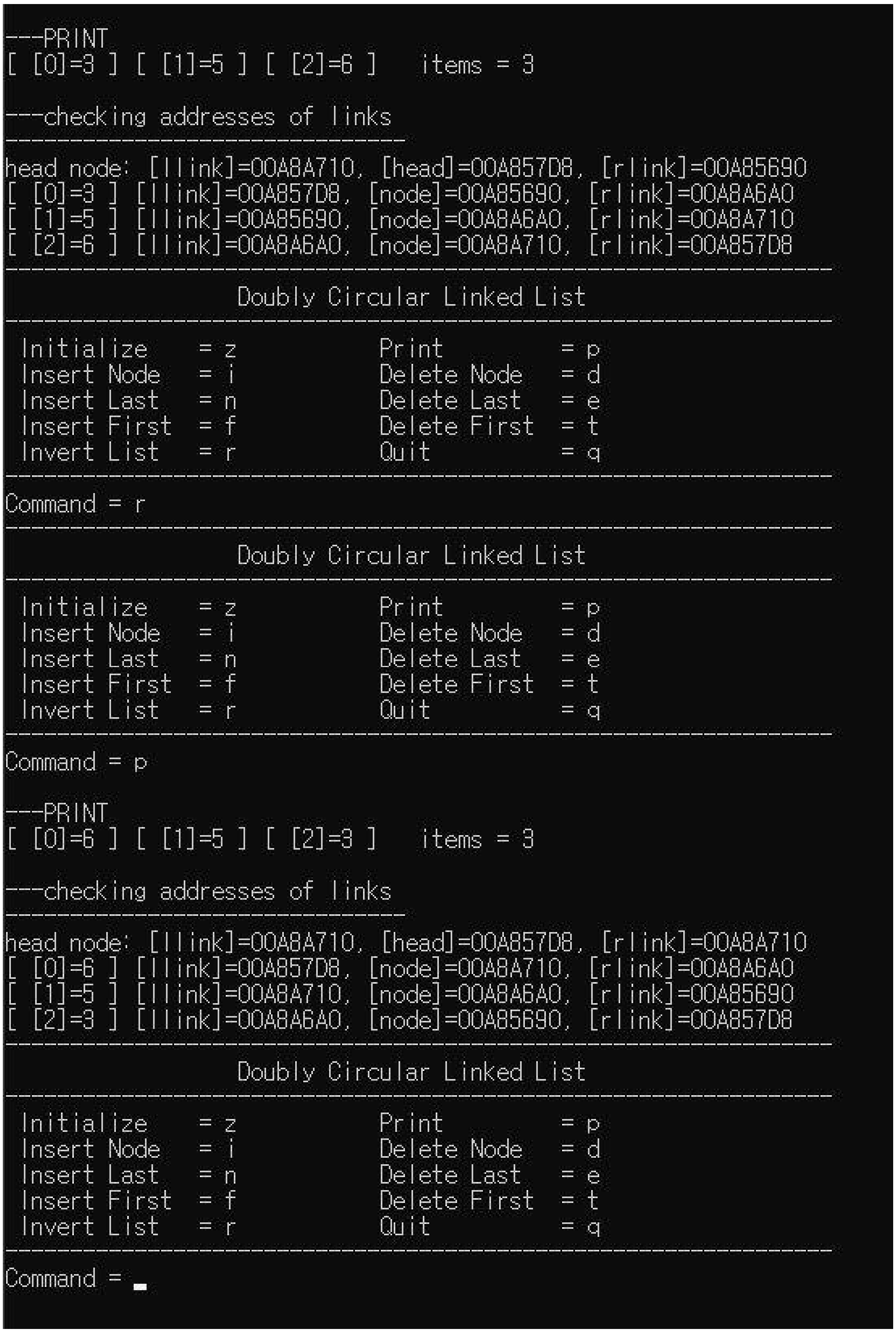
```
---PRINT
[ [0]=3 ] [ [1]=5 ] [ [2]=7 ]   items = 3

---checking addresses of links
-------------------------------
head node: [llink]=00E6AB88, [head]=00E657D8, [rlink]=00E65690
[ [0]=3 ] [llink]=00E657D8, [node]=00E65690, [rlink]=00E6AB18
[ [1]=5 ] [llink]=00E65690, [node]=00E6AB18, [rlink]=00E6AB88
[ [2]=7 ] [llink]=00E6AB18, [node]=00E6AB88, [rlink]=00E657D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
----------------------------------------------------------------
Command = t
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
----------------------------------------------------------------
Command = p

---PRINT
[ [0]=5 ] [ [1]=7 ]   items = 2

---checking addresses of links
-------------------------------
head node: [llink]=00E6AB88, [head]=00E657D8, [rlink]=00E6AB18
[ [0]=5 ] [llink]=00E657D8, [node]=00E6AB18, [rlink]=00E6AB88
[ [1]=7 ] [llink]=00E6AB18, [node]=00E6AB88, [rlink]=00E657D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
----------------------------------------------------------------
Command = e
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
----------------------------------------------------------------
Command = p

---PRINT
[ [0]=5 ]   items = 1
```



```
[ [0]=5 ]   items = 1

---checking addresses of links
-------------------------------
head node: [llink]=00E6AB18, [head]=00E657D8, [rlink]=00E6AB18
[ [0]=5 ] [llink]=00E657D8, [node]=00E6AB18, [rlink]=00E657D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = d
Your Key = 5
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = p

---PRINT
  items = 0

---checking addresses of links
-------------------------------
head node: [llink]=00E657D8, [head]=00E657D8, [rlink]=00E657D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = i2i7i5
Your Key = -------------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = Your Key = ---------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
```



```
Command = Your Key = ----------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------------
Command = p

---PRINT
[ [0]=2 ] [ [1]=5 ] [ [2]=7 ]   items = 3

---checking addresses of links
-------------------------------------
head node: [llink]=00E65D58, [head]=00E657D8, [rlink]=00E65690
[ [0]=2 ] [llink]=00E657D8, [node]=00E65690, [rlink]=00E604D0
[ [1]=5 ] [llink]=00E65690, [node]=00E604D0, [rlink]=00E65D58
[ [2]=7 ] [llink]=00E604D0, [node]=00E65D58, [rlink]=00E657D8
-----------------------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------------
Command = d
Your Key = 5
-----------------------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------------
Command = p

---PRINT
[ [0]=2 ] [ [1]=7 ]   items = 2

---checking addresses of links
-------------------------------------
head node: [llink]=00E65D58, [head]=00E657D8, [rlink]=00E65690
[ [0]=2 ] [llink]=00E657D8, [node]=00E65690, [rlink]=00E65D58
[ [1]=7 ] [llink]=00E65690, [node]=00E65D58, [rlink]=00E657D8
-----------------------------------------------------------------------
                    Doubly Circular Linked List
-----------------------------------------------------------------------
 Initialize    = z           Print          = p
 Insert Node   = i           Delete Node    = d
 Insert Last   = n           Delete Last    = e
 Insert First  = f           Delete First   = t
 Invert List   = r           Quit           = q
-----------------------------------------------------------------------
Command =
```



```
---PRINT
[ [0]=3 ] [ [1]=5 ] [ [2]=6 ]   items = 3

---checking addresses of links
-------------------------------
head node: [llink]=00A8A710, [head]=00A857D8, [rlink]=00A85690
[ [0]=3 ] [llink]=00A857D8, [node]=00A85690, [rlink]=00A8A6A0
[ [1]=5 ] [llink]=00A85690, [node]=00A8A6A0, [rlink]=00A8A710
[ [2]=6 ] [llink]=00A8A6A0, [node]=00A8A710, [rlink]=00A857D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = r
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = p

---PRINT
[ [0]=6 ] [ [1]=5 ] [ [2]=3 ]   items = 3

---checking addresses of links
-------------------------------
head node: [llink]=00A8A710, [head]=00A857D8, [rlink]=00A8A710
[ [0]=6 ] [llink]=00A857D8, [node]=00A8A710, [rlink]=00A8A6A0
[ [1]=5 ] [llink]=00A8A710, [node]=00A8A6A0, [rlink]=00A85690
[ [2]=3 ] [llink]=00A8A6A0, [node]=00A85690, [rlink]=00A857D8
----------------------------------------------------------------
                  Doubly Circular Linked List
----------------------------------------------------------------
 Initialize    = z           Print         = p
 Insert Node   = i           Delete Node   = d
 Insert Last   = n           Delete Last   = e
 Insert First  = f           Delete First  = t
 Invert List   = r           Quit          = q
----------------------------------------------------------------
Command = 
```