



ใบงานที่ 10

เรื่อง Doubly Circular Linkedlist

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นายกฤษฎา วิทยา

65543206041-7

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- แสดงโค้ดโปรแกรมเป็นส่วนๆพร้อมทั้งอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นตอนการทดลอง

```
8 #include <stdio.h> //use printf
9 #include <conio.h> //use getch
10 #include <stdlib.h> //use malloc
11 #define HeadInfo -999 // Define data of Head Node
12
13 typedef struct Node { // Declare structure of node
14     int info;
15     struct Node *llink;
16     struct Node *rlink;
17 } Node;
18
19 struct Node *H, *H1, *p, *q; // Declare pointer node
20 int i, j, k, n, data;
21 char ch;
```

#include <stdio.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการทำงานกับข้อมูลและการแสดงผลบนหน้าจอ

#include <conio.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการทำงานที่เกี่ยวข้องกับคีย์บอร์ด

#include <stdlib.h>: เป็นการเรียกใช้ไลบรารีของภาษา C สำหรับการจัดการกับหน่วยความจำ (memory allocation) และฟังก์ชันที่เกี่ยวข้อง

#define HeadInfo -999: กำหนดค่าคงที่ HeadInfo เป็น -999 ซึ่งจะใช้เป็นข้อมูลพิเศษในโหนดแรก (HEAD NODE)

typedef struct Node: ประกาศโครงสร้างข้อมูล Node ที่มี info เป็นตัวแปรเก็บข้อมูล และ llink, rlink เป็นตัวแปรเก็บพอยน์เตอร์ที่เชื่อมโยงกับโหนดก่อนหน้าและถัดไปในรายการ

struct Node *H, *H1, *p, *q; ประกาศตัวแปรพอยน์เตอร์ที่ใช้ในโปรแกรม

```
23 Node *Allocate() { // Allocate 1 node from storage pool
24     struct Node *temp;
25     temp = (Node *)malloc(sizeof(Node)); // Allocate node by size declare
26     return (temp);
27 }
```

Node *Allocate(): ฟังก์ชัน Allocate ใช้สำหรับจองพื้นที่หน่วยความจำสำหรับโหนดใหม่ โดยใช้ malloc เพื่อจองพื้นที่ขนาด sizeof(Node) และส่งกลับเป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังโหนดใหม่

```

29 void CreateNNode(int n) { // Create N Node put data and link its
30     int i, temp;
31     H1 = H; // Start H1 at here
32     for (i = 1; i ≤ n; i++) { // Count N Node
33         p = Allocate(); // Allocate New Node
34         temp = 1 + rand() % 99; // random difference number 1..99
35         p→info = temp; // Put random data in to node
36         H1→rlink = p; // Link first node to second node
37         p→llink = H1; // LLink point back to predecessor node
38         H1 = p; // Let H1 point to last node
39         H1→rlink = H; // Set rlink of H1 point to HEAD NODE
40         H→llink = H1; // Set LLink of H point to H1
41     }
42 }

```

void CreateNNode(int n): ฟังก์ชัน CreateNNode ใช้สำหรับสร้าง Doubly Circular Linked List โดยรับจำนวน n โหนดที่ต้องการสร้าง และใส่ข้อมูลที่สุ่มได้ในแต่ละโหนด โดยเริ่มต้นที่โหนด HEAD

```

44 void ShowAllNode() {
45     printf("H = %x\n", H); // Display address of pointer H
46     p = H; // Set start point of p pointer
47     i = 1; // set start value of counter
48     if (p→rlink ≠ H) { // if have more node
49         p = p→rlink; // Skip pointer to first node
50         while (p ≠ H) { // While if P ≠ H
51             printf("%d) : %x\t", i, p); // Show COUNTER and POINTER
52             printf("LLINK : %x\t", p→llink); // Show LLink
53             printf("INFO : %d\t", p→info); // Show INFO
54             printf("RLINK : %x\n", p→rlink); // Show RLink
55             p = p→rlink; // Skip to next node
56             i++; // Skip Counter
57         } // End While
58     } // End if
59 } // End Fn.

```

void ShowAllNode(): ฟังก์ชัน ShowAllNode ใช้สำหรับแสดงข้อมูลในโหนดทั้งหมดในรายการ โดยเริ่มต้นที่โหนด HEAD และทำการวนลูปผ่านโหนดทั้งหมดในรายการ แสดงข้อมูลในแต่ละโหนด และลิงก์ที่ชี้ไปยังโหนดถัดไปและโหนดก่อนหน้า

```

61 void InsertAfter(int data1) {
62     int temp; // Temporary variable
63     if (H→rlink == H)
64         printf("Linked List have no node!!..\n");
65     else {
66         H1 = H→rlink; // Let H1 point at 1st node
67         while (H1→info ≠ HeadInfo) { // Search for the data while H1 loop back to HAED Node
68             if (H1→info == data1) { // if Found
69                 p = Allocate(); // Allocat one node from storage pool
70                 printf("\nInsert data : "); // Input data for insert
71                 scanf("%d", &temp); // Read from KBD
72                 p→info = temp; // Entry temporary data into INFO of node
73                 if (H1→rlink == H) { // IF H1 is Last Node
74                     p→rlink = H; // Let p Point to HEAD Node
75                     H→llink = p; // Let H Point to Last Node
76                 } else {
77                     p→rlink = H1→rlink; // Change pointer 1st for insert node (FAR to NEAR)
78                     H1→rlink→llink = p; // LLink(RLink(H1))=p
79                 }
80                 p→llink = H1; // LLink(P)=H1
81                 H1→rlink = p; // RLink(H1)=p101
82             } // End if
83             H1 = H1→rlink; // Skip H1 to next node
84         } // End while
85     } // End IF
86 } // End Fn.

```

void InsertAfter(int data1): ฟังก์ชัน InsertAfter ใช้สำหรับแทรกโหนดใหม่หลังจากโหนดที่มีข้อมูลเท่ากับ data1 โดยให้ผู้ใช้งานป้อนข้อมูลใหม่ที่ต้องการแทรกหลังจากนั้นสร้างโหนดใหม่และแก้ไขลิงก์ให้เป็นอย่างถูกต้อง

```
88 void InsertBefore(int data1) {
89     int temp; // Temporary variable
90     if (H->rlink == H)
91         printf("Linked List have no node!!..\n");
92     else {
93         H1 = H->rlink; // Let H1 point at 1st node
94         while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
95             if (H1->info == data1) { // If Found
96                 p = Allocate(); // Allocate one node from storage pool
97                 printf("\nInsert data : "); // Input data for insert
98                 scanf("%d", &temp); // Read from KBD
99                 p->info = temp; // Entry temporary data into INFO of node
100             if (H1->llink == H) { // First Node
101                 p->llink = H; // LLINK(p) Point to HEAD Node
102                 H->rlink = p; // RLINK(H) Point to p
103             } else {
104                 H1->llink->rlink = p; // RLINK(LLINK(H1))=p
105                 p->llink = H1->llink; // LLINK(p)=LLINK(H1)
106             }
107             H1->llink = p; // LLINK(H1)=p
108             p->rlink = H1; // RLINK(p)=H1
109         } // End if
110         H1 = H1->rlink; // Skip H1 to next node
111     } // End while
112 } // End IF
113 } // End Fn
```

void InsertBefore(int data1): ฟังก์ชัน InsertBefore ใช้สำหรับแทรกโหนดใหม่ก่อนโหนดที่มีข้อมูลเท่ากับ data1 โดยให้ผู้ใช้งานป้อนข้อมูลใหม่ที่ต้องการแทรกก่อนหน้านั้นและสร้างโหนดใหม่และแก้ไขลิงก์ให้เป็นอย่างถูกต้อง

```
115 void DeleteBefore(int data1) {
116     int temp; // Temporary variable
117     if (H->rlink == H)
118         printf("Linked List have NO NODE!!..\n");
119     else {
120         H1 = H->rlink; // Let H1 point at 1st node
121         while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
122             if (H1->info == data1) { // If Found
123                 if (H1->llink == H) // If no more node
124                     printf("No more node before here,Can't delete it!!!\n");
125                 else {
126                     p = H1->llink; // Mark at node for Delete
127                     H1->llink = p->llink; // If not set link of H1 point same address of p
128                     p->llink->rlink = H1;
129                     free(p); // Free node to storage pool
130                 } // End if2
131             } // End if1
132             H1 = H1->rlink; // Skip H1 to next node
133         } // End while
134     } // End IF
135 } // End Fn.
```

void DeleteBefore(int data1): ฟังก์ชัน DeleteBefore ใช้สำหรับลบโหนดก่อนโหนดที่มีข้อมูลเท่ากับ data1 โดยลบโหนดนั้นออกจาก Doubly Circular Linked List ถ้าไม่มีโหนดก่อนหน้า จะแสดงข้อความแจ้งเตือนว่าไม่สามารถลบได้

```

137 void DeleteSelf(int data1) {
138     int temp; // Temporary variable
139     if (H->rlink == H)
140         printf("Linked List have NO NODE!!..\n");
141     else {
142         H1 = H->rlink; // Let H1 point at 1st node
143         while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
144             if (H1->info == data1) { // If Found
145                 p = H1; // Mark at node for Delete
146                 if (p->llink == H && p->rlink == H) { // If only one node
147                     H->llink = H; // Set HEAD Pointer point it self
148                     H->rlink = H;
149                 } else {
150                     p->llink->rlink = p->rlink;
151                     p->rlink->llink = p->llink;
152                 }
153                 free(p); // Free node to storage pool
154             } // End if1
155             H1 = H1->rlink; // Skip H1 to next node
156         } // End while
157     } // End IF
158 } // End Fn.

```

void DeleteBefore(int data1): ฟังก์ชัน DeleteBefore สำหรับลบโหนดก่อนโหนดที่มีข้อมูลที่ตรงกับค่า data1 โดยลบโหนดนั้นออกจาก Doubly Circular Linked List หากไม่มีโหนดก่อนหน้า จะแสดงข้อความแจ้งเตือนว่าไม่สามารถลบได้

```

160 void DeleteAfter(int data1) {
161     int temp; // Temporary variable
162     if (H->rlink == H)
163         printf("Linked List have NO NODE!!..\n");
164     else {
165         H1 = H->rlink; // Let H1 point at 1st node
166         while (H1->info != HeadInfo) { // Search for the data while H1 loop back to HAED Node
167             if (H1->info == data1) { // If Found
168                 if (H1->rlink == H) // If no more node
169                     printf("No more node from here,Can't delete it!!..\n");
170             } else {
171                 p = H1->rlink; // Mark at node for Delete
172                 H1->rlink = p->rlink; // If not set link of H1 point same address of p
173                 p->rlink->llink = H1;
174                 free(p); // Free node to storage pool
175             } // End if2
176         } // End if1
177         H1 = H1->rlink; // Skip H1 to next node
178     } // End while
179 } // End IF
180 } // End Fn

```

void DeleteAfter(int data1): ฟังก์ชันที่ใช้ลบโหนดในลิงค์ลิสต์หลังจากโหนดที่มีข้อมูลเท่ากับ data1 ในกรณีที่ไม่มีโหนดตามหลัง จะแสดงข้อความว่า "No more node from here,Can't delete it!!!"

```

182 int main() // MAIN Fn.
183 {
184     p = Allocate(); // Create HEAD NODE
185     p->info = HeadInfo; // Special data for Head node
186     p->llink = p;
187     p->rlink = p; // Let both Link point to itself
188     H = p; // Let H Point to Head node
189     n = 10; // Set amount of node
190     CreateNNode(n); // Call Fn. Create N nodes
191     printf("PROGRAM DOUBLY CIRCULAR LINKED LIST \n");
192     printf("===== \n");
193     printf("All Data in Linked List \n");
194     ShowAllNode(); // Call Fn. Show all node
195     ch = ' ';
196     while (ch != 'E')
197     {
198         printf("MENU>> [B:InsertBefore A:InsertAfter\n");
199         printf(" 0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]\n");
200         ch = getch();
201         switch (ch)
202         {
203             case 'B':
204                 printf("\nInsert Before data : "); // Input data for insert after
205                 scanf("%d", &data);
206                 InsertBefore(data); // Call Fn. Insert after data
207                 printf("\nAll Data in Linked List AFTER INSERTED\n");
208                 ShowAllNode(); // Call Fn. Show all node
209                 break;
210             case 'A':
211                 printf("\nInsert After data : "); // Input data for insert after
212                 scanf("%d", &data);
213                 InsertAfter(data); // Call Fn. Insert after data
214                 printf("\nAll Data in Linked List AFTER INSERTED\n");
215                 ShowAllNode(); // Call Fn. Show all node
216                 break;
217             case '0':
218                 printf("\nDelete Before data : "); // Input data for Delete after
219                 scanf("%d", &data);
220                 DeleteBefore(data); // Call Fn. Delete after data
221                 printf("\nAll Data in Linked List AFTER DELETED\n");
222                 ShowAllNode(); // Call Fn. Show all node
223                 break;
224             case 'X':
225                 printf("\nDelete ItSelf data : "); // Input data for Delete after
226                 scanf("%d", &data);
227                 DeleteSelf(data); // Call Fn. Delete after data
228                 printf("\nAll Data in Linked List ITSELF DELETED\n");
229                 ShowAllNode(); // Call Fn. Show all node
230                 break;
231             case 'D':
232                 printf("\nDelete After data : "); // Input data forDelete after
233                 scanf("%d", &data);
234                 DeleteAfter(data); // Call Fn. Delete after data
235                 printf("\nAll Data in Linked List AFTER DELETED\n");
236                 ShowAllNode(); // Call Fn. Show all node
237                 break;
238         } // End Switch...case
239     } // End While */
240     return (0);
241 } // End MAIN

```

int main(): ฟังก์ชันหลักที่ใช้เรียกใช้ฟังก์ชันด้านบนเพื่อสร้างและจัดการกับ Doubly Circular Linked List โดยมีเมนูให้เลือกในการเพิ่มหรือลบโหนดจากรายการ และมีการแสดงรายการหลังจากแก้ไขทุกครั้งที่มีการเปลี่ยนแปลง

Run Program

```
PowerShell
kritis D:\WorkAndProject\C\Data_Structures\Week_06 main pwsh
$ .\DoublyCircularLinkedList.exe
PROGRAM DOUBLY CIRCULAR LINKED LIST
=====
All Data in Linked List
H = e40469f0
1) : e4046a18 LLINK : e40469f0 INFO : 42 RLINK : e4046a30
2) : e4046a30 LLINK : e4046a18 INFO : 54 RLINK : e4046a50
3) : e4046a50 LLINK : e4046a30 INFO : 98 RLINK : e4046a70
4) : e4046a70 LLINK : e4046a50 INFO : 68 RLINK : e4046a90
5) : e4046a90 LLINK : e4046a70 INFO : 63 RLINK : e4046ab0
6) : e4046ab0 LLINK : e4046a90 INFO : 83 RLINK : e4046ad0
7) : e4046ad0 LLINK : e4046ab0 INFO : 94 RLINK : e4046af0
8) : e4046af0 LLINK : e4046ad0 INFO : 55 RLINK : e4046b10
9) : e4046b10 LLINK : e4046af0 INFO : 35 RLINK : e4046b30
10) : e4046b30 LLINK : e4046b10 INFO : 12 RLINK : e40469f0
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]

Insert Before data : 42

Insert data : 20

All Data in Linked List AFTER INSERTED
H = e40469f0
1) : e4046b50 LLINK : e40469f0 INFO : 20 RLINK : e4046a18
2) : e4046a18 LLINK : e4046b50 INFO : 42 RLINK : e4046a30
3) : e4046a30 LLINK : e4046a18 INFO : 54 RLINK : e4046a50
4) : e4046a50 LLINK : e4046a30 INFO : 98 RLINK : e4046a70
5) : e4046a70 LLINK : e4046a50 INFO : 68 RLINK : e4046a90
6) : e4046a90 LLINK : e4046a70 INFO : 63 RLINK : e4046ab0
7) : e4046ab0 LLINK : e4046a90 INFO : 83 RLINK : e4046ad0
8) : e4046ad0 LLINK : e4046ab0 INFO : 94 RLINK : e4046af0
9) : e4046af0 LLINK : e4046ad0 INFO : 55 RLINK : e4046b10
10) : e4046b10 LLINK : e4046af0 INFO : 35 RLINK : e4046b30
11) : e4046b30 LLINK : e4046b10 INFO : 12 RLINK : e40469f0
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
```

Insert Before

```
PowerShell
Insert data : 20 om here,Can't delete it!!!\n");
All Data in Linked List AFTER INSERTED
H = ba4769f0
1) : ba476b50 LLINK : ba4769f0 INFO : 20 RLINK : ba476a18
2) : ba476a18 LLINK : ba476b50 INFO : 42 RLINK : ba476a30
3) : ba476a30 LLINK : ba476a18 INFO : 54 RLINK : ba476a50
4) : ba476a50 LLINK : ba476a30 INFO : 98 RLINK : ba476a70
5) : ba476a70 LLINK : ba476a50 INFO : 68 RLINK : ba476a90
6) : ba476a90 LLINK : ba476a70 INFO : 63 RLINK : ba476ab0
7) : ba476ab0 LLINK : ba476a90 INFO : 83 RLINK : ba476ad0
8) : ba476ad0 LLINK : ba476ab0 INFO : 94 RLINK : ba476af0
9) : ba476af0 LLINK : ba476ad0 INFO : 55 RLINK : ba476b10
10) : ba476b10 LLINK : ba476af0 INFO : 35 RLINK : ba476b30
11) : ba476b30 LLINK : ba476b10 INFO : 12 RLINK : ba4769f0
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]

Insert After data : 42

Insert data : 21

All Data in Linked List AFTER INSERTED
H = ba4769f0
1) : ba476b50 LLINK : ba4769f0 INFO : 20 RLINK : ba476a18
2) : ba476a18 LLINK : ba476b50 INFO : 42 RLINK : ba476b70
3) : ba476b70 LLINK : ba476a18 INFO : 21 RLINK : ba476a30
4) : ba476a30 LLINK : ba476b70 INFO : 54 RLINK : ba476a50
5) : ba476a50 LLINK : ba476a30 INFO : 98 RLINK : ba476a70
6) : ba476a70 LLINK : ba476a50 INFO : 68 RLINK : ba476a90
7) : ba476a90 LLINK : ba476a70 INFO : 63 RLINK : ba476ab0
8) : ba476ab0 LLINK : ba476a90 INFO : 83 RLINK : ba476ad0
9) : ba476ad0 LLINK : ba476ab0 INFO : 94 RLINK : ba476af0
10) : ba476af0 LLINK : ba476ad0 INFO : 55 RLINK : ba476b10
11) : ba476b10 LLINK : ba476af0 INFO : 35 RLINK : ba476b30
12) : ba476b30 LLINK : ba476b10 INFO : 12 RLINK : ba4769f0
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
```

Insert After

```
PowerShell x + v - □ x
Insert After data : 42
Insert data : 21
All Data in Linked List AFTER INSERTED
H = ba4769f8
1) : ba476b58 LLINK : ba4769f8 INFO : 20 RLINK : ba476a18
2) : ba476a18 LLINK : ba476b58 INFO : 42 RLINK : ba476b78
3) : ba476b78 LLINK : ba476a18 INFO : 21 RLINK : ba476a38
4) : ba476a38 LLINK : ba476b78 INFO : 54 RLINK : ba476a58
5) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
6) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476a98
7) : ba476a98 LLINK : ba476a78 INFO : 63 RLINK : ba476ab8
8) : ba476ab8 LLINK : ba476a98 INFO : 83 RLINK : ba476ad8
9) : ba476ad8 LLINK : ba476ab8 INFO : 94 RLINK : ba476af8
10) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
11) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
12) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
Delete Before data : 42
All Data in Linked List AFTER DELETED
H = ba4769f8
1) : ba476a18 LLINK : ba4769f8 INFO : 42 RLINK : ba476b78
2) : ba476b78 LLINK : ba476a18 INFO : 21 RLINK : ba476a38
3) : ba476a38 LLINK : ba476b78 INFO : 54 RLINK : ba476a58
4) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
5) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476a98
6) : ba476a98 LLINK : ba476a78 INFO : 63 RLINK : ba476ab8
7) : ba476ab8 LLINK : ba476a98 INFO : 83 RLINK : ba476ad8
8) : ba476ad8 LLINK : ba476ab8 INFO : 94 RLINK : ba476af8
9) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
10) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
11) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
```

Delete Before

```
PowerShell x + v - □ x
11) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
12) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
Delete Before data : 42
All Data in Linked List AFTER DELETED
H = ba4769f8
1) : ba476a18 LLINK : ba4769f8 INFO : 42 RLINK : ba476b78
2) : ba476b78 LLINK : ba476a18 INFO : 21 RLINK : ba476a38
3) : ba476a38 LLINK : ba476b78 INFO : 54 RLINK : ba476a58
4) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
5) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476a98
6) : ba476a98 LLINK : ba476a78 INFO : 63 RLINK : ba476ab8
7) : ba476ab8 LLINK : ba476a98 INFO : 83 RLINK : ba476ad8
8) : ba476ad8 LLINK : ba476ab8 INFO : 94 RLINK : ba476af8
9) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
10) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
11) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
Delete ItSelf data : 63
All Data in Linked List ITSELF DELETED
H = ba4769f8
1) : ba476a18 LLINK : ba4769f8 INFO : 42 RLINK : ba476b78
2) : ba476b78 LLINK : ba476a18 INFO : 21 RLINK : ba476a38
3) : ba476a38 LLINK : ba476b78 INFO : 54 RLINK : ba476a58
4) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
5) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476a98
6) : ba476a98 LLINK : ba476a78 INFO : 83 RLINK : ba476ad8
7) : ba476ad8 LLINK : ba476a98 INFO : 94 RLINK : ba476af8
8) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
9) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
10) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
```

Delete Itself


```
PowerShell x + -
10) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
11) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]

Delete ItSelf data : 43

All Data in Linked List ITSELF DELETED
H = ba4769f8
1) : ba476a18 LLINK : ba4769f8 INFO : 42 RLINK : ba476b78
2) : ba476b78 LLINK : ba476a18 INFO : 21 RLINK : ba476a38
3) : ba476a38 LLINK : ba476b78 INFO : 54 RLINK : ba476a58
4) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
5) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476ab8
6) : ba476ab8 LLINK : ba476a78 INFO : 83 RLINK : ba476ad8
7) : ba476ad8 LLINK : ba476ab8 INFO : 94 RLINK : ba476af8
8) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
9) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
10) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]

Delete After data : 42

All Data in Linked List AFTER DELETED
H = ba4769f8
1) : ba476a18 LLINK : ba4769f8 INFO : 42 RLINK : ba476a38
2) : ba476a38 LLINK : ba476a18 INFO : 54 RLINK : ba476a58
3) : ba476a58 LLINK : ba476a38 INFO : 98 RLINK : ba476a78
4) : ba476a78 LLINK : ba476a58 INFO : 68 RLINK : ba476ab8
5) : ba476ab8 LLINK : ba476a78 INFO : 83 RLINK : ba476ad8
6) : ba476ad8 LLINK : ba476ab8 INFO : 94 RLINK : ba476af8
7) : ba476af8 LLINK : ba476ad8 INFO : 55 RLINK : ba476b18
8) : ba476b18 LLINK : ba476af8 INFO : 35 RLINK : ba476b38
9) : ba476b38 LLINK : ba476b18 INFO : 12 RLINK : ba4769f8
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
MENU>> [B:InsertBefore A:InsertAfter
0:DeleteBefore X:Deleteitself D:DeleteAfter E:Exit]
```

Delete After

สรุปผลการทดลอง

โปรแกรมนี้ช่วยให้ผู้ใช้สามารถสร้าง Doubly Circular Linked List, แสดงข้อมูลในโนหนดทั้งหมด, แทรกโนหนดใหม่หลังจากโนหนดที่ต้องการ, ลบโนหนดที่ตั้งหลังจากโนหนดที่ต้องการ และทำการออกจากโปรแกรมเมื่อต้องการสิ้นสุดการทำงาน