

연결리스트 실습 코드 그림으로 설명

목차

- 전체 코드
- 그림으로 설명

전체 코드

```
#define CRT SECURE NO WARRIENGS
     Finclude «stdio.h»
     Finclude cstdlib.bo
     // 연결 리스트의 노트를 나타내는 구조체
     typedef struct node
 11
        int data:
        struct mode* next;
110
     3 NOOE:
11
     // 새로운 노드를 정성하는 함수
     NODE* create_node(int value)
14
15
        NODE* new_node = (NODE*)malloc(sizeof(NODE));
3.15
37
        1f (new_node == NULL)
18
            printf("明显引 整容 오류\n");
10
20
            exit(EXIT FAILURE):
21
22
        new_node->data = value;
23
        new node->next = MULL:
24
25
        return new node;
26
27
```

```
77 연결 건스트에 노드를 추가하는 함수
     void insert_node(NODE** head, int value)
10
        NODE* new node = create node(value);
31
32
        if (*head == NULL)
3.3
3.4
35
            // 리스트가 비여인을 경우
96
             *head = new_node;
37
311
        else-
310
40
            77 리스트가 비어인지 않을 경우
41
            MODE* temp - *head;
4.3
            while (temp->next (= NULL)
43
4.0
                temp = temp->next;
45
46
            temp->next = new node;
47
40
49
     // 연결 리스트의 노트를 출력하는 함수
9.3
     void print list(NODE* head)
52
53
        NODE* temp = head;
        while (temp != 10ULL):
54
55
            printf("%d -> ", temp->data);
56
57
            temp = temp->next;
5.0
        printf("MALL\n");
50
641
```

0.1

```
// 메모리 하게 함수
     void free list(NODE* head)
64
65
        NCDE* temp;
66
         while (hend I= HULL)
67
68
            temp = head;
6.0
            head = head->next;
70
            free(temp);
71
73
        printf("明显证 胡胡 些罪\n");
73
74
25
     int main()
26
        NOOR" my_list = NALL;
77
78
        77 연결 리스트에 노드 추가
70
00
        insert node(Key list, 10);
         insert_node(Amy_list, 20);
83.
82
         insert_node(&my_list, 30);
83
        insert_node(&my_list, 40);
64.
85
        7/ 연결 리스트 출작
        printf("연결 리스트: ")1
B6
87
        print_list(my_list);
0.0
        // 메오리 해제
89
90
        free list(my_list);
91
92
         return 0:
93
```

```
C (C17 + GNU extensions)
```

```
known limitations
           NODE ALBINDA
           while(head != NULL)
     67
     68
               temp = head;
               head = head->next;
               free(temp);
           printf("메모리 해제 완료₩n");
     73
     74
         int main()
           NODE *my_list = NULL;
     79
           // 연결 리스트에 노드 추가
           insert_node(&my_list, 10);
     81
           insert_node(&my_list, 20);
           insert_node(&my_list, 30);
           // 연결 리스트 출력
           printf("연결 리스트: ");
           meint lint/mo lintly
     ne.
                         Edit this code
ine that just executed
next line to execute
                                Next >
                                        Last >>
```

Stack	Неар	
main		
my_list pointer to NODE		
ote: ? refers to an uninitia		

```
C (C17 + GNU extensions)
```

```
known limitations
           NODE ALBINDA
           while(head != NULL)
     67
     68
               temp = head;
               head = head->next;
               free(temp);
           printf("메모리 해제 완료₩n");
     73
     74
         int main()
   → 76 {
           NODE *my_list = NULL;
     78
     79
           // 연결 리스트에 노드 추가
           insert_node(&my_list, 10);
     81
           insert_node(&my_list, 20);
           insert_node(&my_list, 30);
           // 연결 리스트 출력
           printf("연결 리스트: ");
           meint lint/mo lintly
     ne
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                        Last >>
```

	Stack	Heap	
main			
my_list	pointer to NODE		

```
C (C17 + GNU extensions)
```

```
known limitations
           NODE ALBINDA
           while(head != NULL)
     67
     68
               temp = head;
               head = head->next;
              free(temp);
           printf("메모리 해제 완료₩n");
     73
     74
         int main()
     76
           NODE *my_list = NULL;
     79
           // 연결 리스트에 노드 추가
   → 80
           insert_node(&my_list, 10);
     81
           insert_node(&my_list, 20);
           insert_node(&my_list, 30);
           // 연결 리스트 출력
           printf("연결 리스트: ");
           meint lint/mo lintly
     ne
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Stack	Неар
my_list pointer to NODE NULL (0x0)	
/C++ details: none [default view]	~

```
C (C17 + GNU extensions)
              known limitations
              exit(EXIT_FAILURE);
          new_node->data = value;
          new_node->next = NULL;
     24
          return new_node;
     26
        // 연결 리스트에 노드를 추가하는 함수
     29 void insert_node(NODE **head, int value)
  → 30 {
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
              *head = new_node;
          else
              // 미시트가 비심어지 않은 겨모
                       Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

	Stack	Hea	р
ain			
my_list	pointer to NODE NULL (0x0)		
nsert_node			
head	pointer to NODE*		
value	int ?		
new_node	pointer to NODE		

```
C (C17 + GNU extensions)
              known limitations
              exit(EXIT_FAILURE);
          new_node->data = value;
          new_node->next = NULL;
     24
          return new node;
     26
        // 연결 리스트에 노드를 추가하는 함수
     29 void insert_node(NODE **head, int value)
  → 30 {
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
              *head = new_node;
          else
              // 미시트가 비심어지 않은 겨모
                        Edit this code
ine that just executed
next line to execute
               << First
                      < Prev
```

Next >

Last >>

	Stack	He	ар
nain	= ====		r
my_list	pointer to NODE NULL (0x0)		
nsert_node			
head	pointer to NODE*		
value	int 10		
new_node	pointer to NODE		

```
C (C17 + GNU extensions)
```

```
known limitations
   #THICTURE NOTULINATE
   // 연결 리스트의 노드를 나타내는 구조체
 6 typedef struct node
     int data;
     struct node *next;
10 } NODE:
   // 새로운 노드를 생성하는 함수
13 NODE *create node(int value)
     NODE *new_node = (NODE *)malloc(sizeof(NODE));
16
     if(new_node == NULL)
18
        printf("메모리 할당 오류빿");
        exit(EXIT_FAILURE);
     new_node->data = value;
     new_node->next = NULL;
0.4
```

Edit this code

→ line that just executed
→ next line to execute

```
Print output (drag lower right corner to resize)
                         Stack
                                          Heap
 main
                pointer to NODE
     my_list
                NULL (0x0)
 insert_node
              pointer to NODE*
       head
      value
              10
              pointer to NODE
 new_node
 create_node
       value
                pointer to NODE
  new_node
```

Note: ? refers to an uninitialized value

C/C++ details: none [default view]

```
C (C17 + GNU extensions)
```

```
known limitations
     #THICTURE NOTULINATE
     // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
       int data;
       struct node *next;
  10 } NODE:
     // 새로운 노드를 생성하는 함수
  13 NODE *create node(int value)
→ 14 {
       NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
       if(new_node == NULL)
  18
           printf("메모리 할당 오류빿");
           exit(EXIT_FAILURE);
       new_node->data = value;
       new_node->next = NULL;
  0.4
```

Edit this code

```
< Prev
<< First
                   Next >
                              Last >>
```

ine that just executed next line to execute

Print output (drag lower right corner to resize) Stack Heap main pointer to NODE my_list NULL (0x0) insert_node pointer to NODE* head value 10 pointer to NODE new node create_node value 10 pointer to NODE new_node

Note: ? refers to an uninitialized value C/C++ details: none [default view]

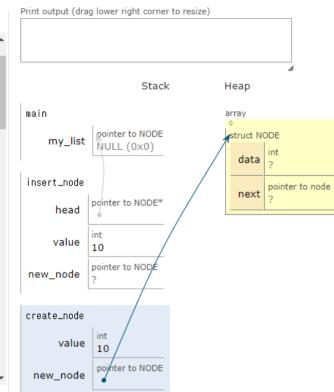
```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
        struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
      NODE *create node(int value)
  14
\rightarrow 15
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
→ 17
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
            exit(EXIT_FAILURE);
  21
        new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

→ line that just executed → next line to execute





Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

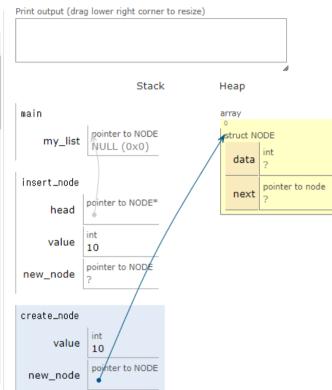
```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
        struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
      NODE *create node(int value)
  14
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
\rightarrow 17
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
            exit(EXIT_FAILURE);
  21
        new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

```
next line to execute

<
```

ine that just executed



Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTUTIONIE
     // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
       struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
     NODE *create node(int value)
  14
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
           exit(EXIT_FAILURE);
  21
\rightarrow 22
        new_node->data = value;
→ 23
        new_node->next = NULL;
  0.4
```

Edit this code

Next >

Last >>

< Prev

ine that just executed next line to execute

<< First

Note
Note
amb

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list NULL (0x0) data 10 insert node pointer to node next pointer to NODE* head value 10 pointer to NODE new node create_node value 10 pointer to NODE new_node

Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
     1 ↔ - €
           NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new_node == NULL)
     18
               printf("메모리 할당 오류₩n");
               exit(EXIT FAILURE);
           new_node->data = value;
   → 23
           new_node->next = NULL;
     24
   → 25
           return new node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
           NODE *new node = create node(value);
           if(*head == NULL)
     34
               // 기치는가 비하이트 겨드
     OE.
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

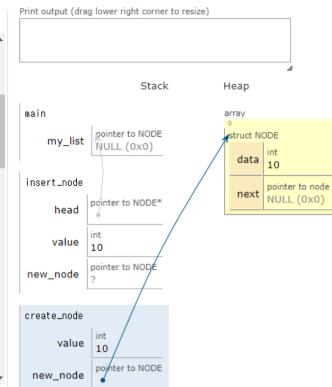
Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list NULL (0x0) data 10 insert node pointer to node next NULL (0x0) pointer to NODE* head value 10 pointer to NODE new node create_node value 10 pointer to NODE new_node

Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
   1 ↔ - €
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
   16
        if(new_node == NULL)
   18
            printf("메모리 할당 오류₩n");
             exit(EXIT FAILURE);
        new_node->data = value;
        new_node->next = NULL;
   24
\rightarrow 25
        return new node;
→ 26
      // 연결 리스트에 노드를 추가하는 함수
       void insert_node(NODE **head, int value)
        NODE *new node = create node(value);
         if(*head == NULL)
   34
             // 기치는가 비하이트 겨드
   OE.
                       Edit this code
```

→ line that just executed
→ next line to execute



Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
      1 ↔ - €
           NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new node == NULL)
     18
     19
               printf("메모리 할당 오류₩n");
               exit(EXIT FAILURE);
           new_node->data = value;
           new_node->next = NULL;
     24
           return new node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
   \rightarrow 31
           NODE *new node = create node(value);
   → 33.
           if(*head == NULL)
     34
      oc.
                // 기치는가 비사이트 경우
                          Edit this code
ine that just executed
next line to execute
```

<< First < Prev Next > Last >> Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list NULL (0x0) data 10 insert node pointer to node next NULL (0x0) pointer to NODE head value 10 pointer to NODE new node

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view]

```
C (C17 + GNU extensions)
```

```
known limitations
        recurr new_noue/
  26
      // 연결 리스트에 노드를 추가하는 함수
      void insert node(NODE **head, int value)
        NODE *new_node = create_node(value);
\rightarrow 33
        if(*head == NULL)
           // 리스트가 비어있을 경우
→ 36
           *head = new node;
        else
  40
           // 리스트가 비어있지 않을 경우
  41
           NODE *temp = *head;
            while(temp->next != NULL)
  44
                temp = temp->next;
  45
   40
            tome Secut - sem sade:
                      Edit this code
```

→ line that just executed
→ next line to execute

<< First | < Prev | Next > Last >>

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list NULL (0x0) data 10 insert node pointer to node next NULL (0x0) pointer to NODE head value 10 pointer to NODE new node

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view]
▼

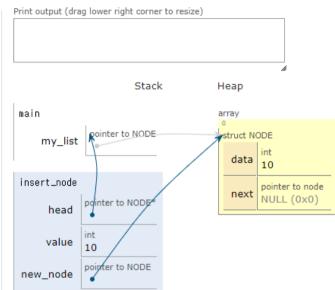
C (C17 + GNU extensions) known limitations

```
0.1
         - 3
       else
  40
           // 리스트가 비어있지 않을 경우
  41
           NODE *temp = *head;
           while(temp->next != NULL)
  43
  44
               temp = temp->next;
  45
  46
           temp->next = new_node;
  47
→ 48
  49
      // 연결 리스트의 노드를 출력하는 함수
     void print_list(NODE *head)
  52
       NODE *temp = head;
       while(temp != NULL)
  55
           printf("%d -> ", temp->data);
           temp = temp->next;
         ٦
  En
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view] ✓

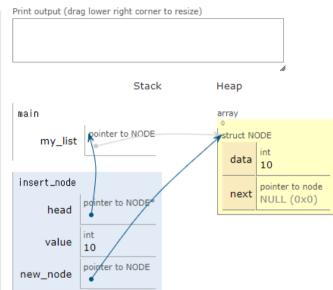
C (C17 + GNU extensions) known limitations

```
0.1
         .
        else
  40
           // 리스트가 비어있지 않을 경우
  41
           NODE *temp = *head;
           while(temp->next != NULL)
  43
  44
               temp = temp->next;
  45
  46
           temp->next = new_node;
  47
48
  49
      // 연결 리스트의 노드를 출력하는 함수
      void print_list(NODE *head)
  52
       NODE *temp = head;
        while(temp != NULL)
  55
           printf("%d -> ", temp->data);
           temp = temp->next;
         ٦
  En
```

Edit this code

→ line that just executed→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view] ✓

```
C (C17 + GNU extensions)
```

```
known limitations
              TICC(COMP)
          printf("메모리 해제 완료₩n");
     73
     74
        int main()
     76
          NODE *my_list = NULL;
     78
          // 연결 리스트에 노드 추가
  → 80
          insert_node(&my_list, 10);
  → 81
          insert_node(&my_list, 20);
          insert_node(&my_list, 30);
     84
          // 연결 리스트 출력
          printf("연결 리스트: ");
          print_list(my_list);
          // 메모리 해제
          free_list(my_list);
           estues O:
                        Edit this code
ine that just executed
```

→ next line to execute

st >>

Print output (drag lo	wer right corner t	o resize)	
	Stack	Неар	
main		array	
my_list pointer	to NODE	struct N	ODE
,		data	int 10
		next	pointer to not NULL (0x0)

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view] V

```
C (C17 + GNU extensions)
              known limitations
              exit(EXIT_FAILURE);
          new node->data = value;
          new node->next = NULL;
     24
          return new node;
     26
        // 연결 리스트에 노드를 추가하는 함수
        void insert node(NODE **head, int value)
  → 30
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new node;
     37
          else
     40
              // 미스트가 비전이지 않은 겨드
                        Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list data 10 insert node pointer to node next NULL (0x0) pointer to NODE* head value pointer to NODE new node

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view]

```
C (C17 + GNU extensions)
              known limitations
               exit(EXIT_FAILURE);
           new node->data = value;
           new node->next = NULL;
     24
           return new node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert node(NODE **head, int value)
  \rightarrow 30
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new node;
     37
           else
      40
               // 미스트가 비전이지 않은 겨드
                        Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list data 10 insert node pointer to node next NULL (0x0) pointer to NODE* head value 20 pointer to NODE new node

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view]

main

C (C17 + GNU extensions)

```
known limitations
   #THICTURE NOTUTIONIE
   // 연결 리스트의 노드를 나타내는 구조체
 6 typedef struct node
     int data;
     struct node *next;
10 } NODE:
   // 새로운 노드를 생성하는 함수
13 NODE *create node(int value)
     NODE *new_node = (NODE *)malloc(sizeof(NODE));
16
     if(new_node == NULL)
18
19
        printf("메모리 할당 오류\");
         exit(EXIT_FAILURE);
21
     new_node->data = value;
     new_node->next = NULL;
0.4
```

Edit this code

Next >

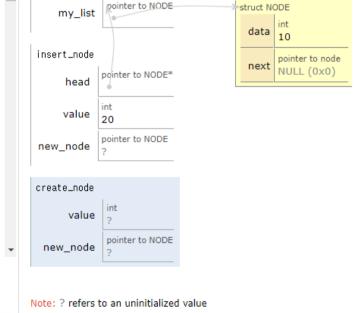
Last >>

< Prev

```
next line to execute
```

<< First

ine that just executed



Stack

Heap

array

Print output (drag lower right corner to resize)

C (C17 + GNU extensions)

```
known limitations
      #THICTURE NATULIDADA
     // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
       int data;
       struct node *next;
  10 } NODE;
     // 새로운 노드를 생성하는 함수
  13 NODE *create_node(int value)
→ 14 {
       NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
       if(new node == NULL)
  19
           printf("메모리 할당 오류빿");
           exit(EXIT_FAILURE);
       new_node->data = value;
       new node->next = NULL;
  0.4
```

Edit this code

→ line that just executed
→ next line to execute



Print output (dra	g lower right corner	to resize)		
	Stack	: 1	Неар	
main			array	
	pointer to NODE		o struct No	ODE
my_list			data	int 10
insert_node			next	pointer to node
head	pointer to NODE*		next	NULL (0x0)
value	int 20			
new_node	pointer to NODE ?			
create_node				
value	int 20			
new_node	pointer to NODE			

Note: ? refers to an uninitialized value

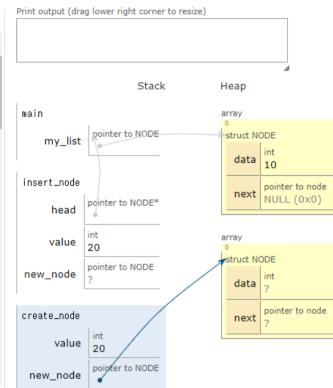
C (C17 + GNU extensions)

```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
        struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
      NODE *create node(int value)
  14
\rightarrow 15
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
→ 17
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
            exit(EXIT_FAILURE);
  21
        new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: ? refers to an uninitialized value

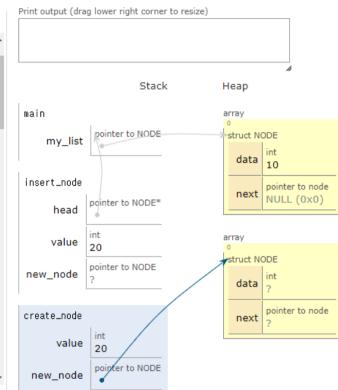
```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
       struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
     NODE *create node(int value)
  14
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
\rightarrow 17
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
           exit(EXIT_FAILURE);
  21
        new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: ? refers to an uninitialized value

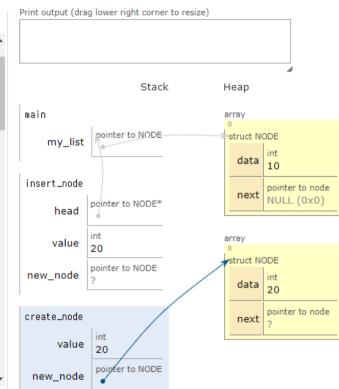
C (C17 + GNU extensions)

```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
       struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
     NODE *create node(int value)
  14
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
           exit(EXIT_FAILURE);
  21
\rightarrow 22
        new_node->data = value;
→ 23
        new_node->next = NULL;
  0.4
```

Edit this code



ine that just executed



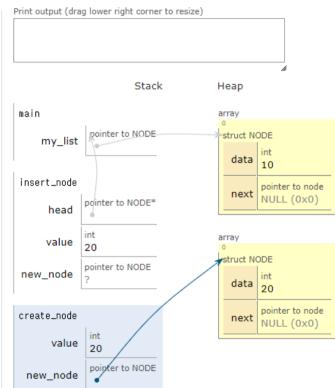
Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
  1 ↔ - €
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
        if(new node == NULL)
  18
           printf("메모리 할당 오류₩n");
            exit(EXIT FAILURE);
        new_node->data = value;
→ 23
        new_node->next = NULL;
  24
→ 25
        return new node;
  26
      // 연결 리스트에 노드를 추가하는 함수
      void insert_node(NODE **head, int value)
        NODE *new node = create node(value);
        if(*head == NULL)
  34
            // 기치는가 비하이트 겨드
  OE.
                     Edit this code
```

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>



Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
                       known limitations
   1 ↔ - €
         NODE *new_node = (NODE *)malloc(sizeof(NODE));
   16
         if(new node == NULL)
   18
             printf("메모리 할당 오류₩n");
             exit(EXIT FAILURE);
         new_node->data = value;
         new_node->next = NULL;
   24
\rightarrow 25
         return new node;
→ 26
       // 연결 리스트에 노드를 추가하는 함수
       void insert_node(NODE **head, int value)
         NODE *new node = create node(value);
         if(*head == NULL)
```

→ line that just executed
→ next line to execute

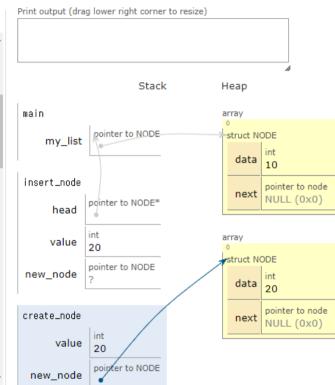
34

OE.

<< First < Prev Next > Last >>

Edit this code

// 기치는가 비하이트 겨드



Note: ? refers to an uninitialized value

```
C (C17 + GNU extensions)
```

```
known limitations
     1 ↔ - €
           NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new node == NULL)
     18
     19
               printf("메모리 할당 오류빿");
               exit(EXIT FAILURE);
           new_node->data = value;
           new_node->next = NULL;
     24
           return new node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
  \rightarrow 31
           NODE *new node = create node(value);
   → 33.
           if(*head == NULL)
     34
     OE.
               // 기치는가 비하이트 겨드
                         Edit this code
ine that just executed
```

→ line that just executed
→ next line to execute

```
<< First < Prev Next > Last >>
```

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert node pointer to node next NULL (0x0) pointer to NODE* head arrav value 20 struct NODE pointer to NODE new node data 20 pointer to node NULL (0x0)

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

```
C/C++ details: none [default view]
```

```
C (C17 + GNU extensions)
     known limitations
```

```
JU L
     31
           NODE *new_node = create_node(value);
  \rightarrow 33
           if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new node;
           else
     40
              // 리스트가 비어있지 않을 경우
              NODE *temp = *head;
               while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
     46
               temp->next = new node;
     47
     48
     49
            연결 리스트의 노드를 출력하는 함수
     El maid meint lint(NODE chand)
                         Edit this code
ine that just executed
next line to execute
```

<< First < Prev Next > Last >>

	Stack	 Неар	**
main		array	
my_list	Pointer to NODE	 struct N	ODE
,-		data	int 10
insert_node		next	pointer to node
head	pointer to NODE*	Hext	NULL (0x0)
value	int	array	
	20	 o •struct No	ODE
new_node	pointer to NODE	data	int 20
temp	pointer to NODE	next	pointer to node

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view]

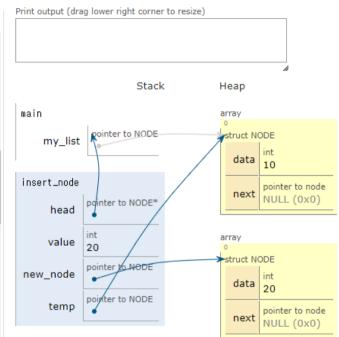
```
C (C17 + GNU extensions)
known limitations
```

```
JU L
        NODE *new_node = create_node(value);
        if(*head == NULL)
  34
  35
            // 리스트가 비어있을 경우
  36
            *head = new node;
        else
  40
            // 리스트가 비어있지 않을 경우
\rightarrow 41
            NODE *temp = *head;
→ 42
            while(temp->next != NULL)
  43
  44
                temp = temp->next;
  45
  46
            temp->next = new node;
  47
  48 }
  49
         연결 리스트의 노드를 출력하는 함수
  E1 would spine that (MODE shood)
                     Edit this code
```

ine that just executed

next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

```
C/C++ details: none [default view] 		✓
```

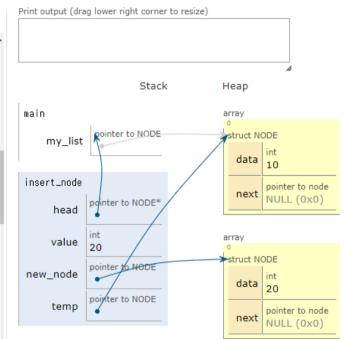
C (C17 + GNU extensions) known limitations

```
JU L
       NODE *new_node = create_node(value);
       if(*head == NULL)
  34
  35
           // 리스트가 비어있을 경우
  36
           *head = new node;
       else
  40
           // 리스트가 비어있지 않을 경우
  41
           NODE *temp = *head;
\rightarrow 42
           while(temp->next != NULL)
  43
  44
               temp = temp->next;
  45
           temp->next = new node;
  47
  48
  49
        연결 리스트의 노드를 출력하는 함수
  El maid meint lint(NODE chand)
                     Edit this code
```

ine that just executed

next line to execute

<< First | < Prev | Next > Last >>



Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

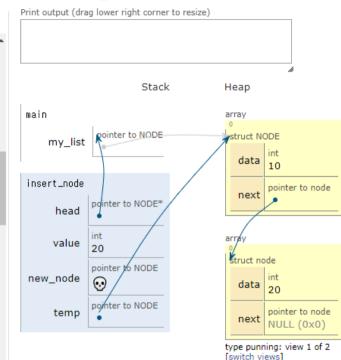
C/C++ details: none [default view]

✓

```
C (C17 + GNU extensions)
known limitations
```

```
JU L
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new node;
           else
     40
              // 리스트가 비어있지 않을 경우
     41
              NODE *temp = *head;
              while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
  → 46
              temp->next = new node;
     47
  → 48
     49
           연결 리스트의 노드를 출력하는 함수
     El maid meint lint(NODE chand)
                        Edit this code
ine that just executed
next line to execute
```

<< First | < Prev | Next > Last >>



Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. locations are

```
C (C17 + GNU extensions)
     known limitations
```

```
JU L
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
              // 리스트가 비어있을 경우
     36
              *head = new node;
          else
              // 리스트가 비어있지 않을 경우
     40
     41
              NODE *temp = *head;
              while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
     46
              temp->next = new node;
     47
  48
     49
           연결 리스트의 노드를 출력하는 함수
     El maid meint lint(NODE chand)
                        Edit this code
ine that just executed
next line to execute
```

<< First < Prev Next > Last >>

Heap array struct NODE
struct NODE
struct NODE
data
10
next pointer to nod
array
struct node
data int 20
next pointer to nod

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. . locations are approximate and may not match the pointer's real address.

```
C (C17 + GNU extensions)
                         known limitations
           printf("메모리 해제 완료₩n");
     73
     74
         int main()
     76
           NODE *mv list = NULL;
     79
           // 연결 리스트에 노드 추가
           insert_node(&my_list, 10);
  \rightarrow 81
           insert node(&mv list, 20);
   → 82
           insert_node(&my_list, 30);
           // 연결 리스트 출력
           printf("연결 리스트: ");
     86
           print_list(my_list);
           // 메모리 해제
           free_list(my_list);
           return O;
     00.1
                         Edit this code
ine that just executed
next line to execute
```

Next >

Last >>

< Prev

<< First

Print output (drag lo		
	Stack	Неар
main		array
my_list pointer	to NODE	struct NODE data int 10 next pointer to noc
		array
		Struct node
		data int 20
		next pointer to not NULL (0x0)

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

```
C/C++ details: none [default view] ✓
```

```
C (C17 + GNU extensions)
              known limitations
              exit(EXIT_FAILURE);
          new_node->data = value;
          new node->next = NULL;
     24
          return new node;
     26
        // 연결 리스트에 노드를 추가하는 함수
     29 void insert node(NODE **head, int value)
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
              *head = new node;
     37
          else
              // 미스트가 비전이지 않은 겨드
                        Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

rint output (dra				
	Stack	: H	Неар	
main		ē	array 0	
my_list	pointer to NODE	3	struct No	int 10
insert_node			next	pointer to nod
head	pointer to NODE*		Hext	<u>^</u>
value	int ?	ā	array	
	pointer to NODE		Struct no	ode
new_node	?		data	int 20
			next	pointer to nod NULL (0x0)

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view] ✓

```
C (C17 + GNU extensions)
              known limitations
              exit(EXIT_FAILURE);
          new node->data = value;
          new node->next = NULL;
     24
     25
          return new node;
     26 }
        // 연결 리스트에 노드를 추가하는 함수
        void insert node(NODE **head, int value)
  → 30 {
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new node;
     37
          else
     40
              // 미스트가 비전이지 않은 겨드
                        Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert node pointer to node next pointer to NODE* head arra value 30 struct node pointer to NODE new node data pointer to node next NULL (0x0)

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

C/C++ details: none [default view] ➤

C (C17 + GNU extensions)

```
known limitations
   #THICTURE NOTULINATE
 5 // 연결 리스트의 노드를 나타내는 구조체
6 typedef struct node
     int data;
     struct node *next;
10 } NODE:
   // 새로운 노드를 생성하는 함수
13 NODE *create node(int value)
     NODE *new_node = (NODE *)malloc(sizeof(NODE));
16
     if(new_node == NULL)
18
        printf("메모리 할당 오류₩n");
        exit(EXIT_FAILURE);
     new_node->data = value;
     new_node->next = NULL;
0.4
```

Edit this code

→ line that just executed → next line to execute



Print output (dra	ig lower right corner	to resize)		
	Stack	: 1	Неар	, ii
main			array	
my_list	pointer to NODE		struct N	DDE
,			data	int 10
insert_node			next	pointer to no
head	pointer to NODE*			<u> </u>
value	int		array	
variae	30		struct no	ode
new_node	pointer to NODE		data	int 20
create_node	I.		next	pointer to no
value	int ?			
new_node	pointer to NODE			

Note: ? refers to an uninitialized value

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members.

```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTUTIONIE
      // 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
        int data;
        struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
      NODE *create node(int value)
\rightarrow 14
        NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
        if(new_node == NULL)
  18
  19
           printf("메모리 할당 오류빿");
            exit(EXIT_FAILURE);
  21
        new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

Next >

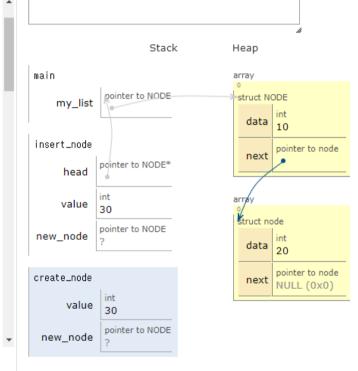
Last >>

< Prev

next line to execute

<< First

ine that just executed



Note: ? refers to an uninitialized value

Print output (drag lower right corner to resize)

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members.

```
C (C17 + GNU extensions)
```

```
known limitations
     #THICTURE NOTULINATE
     기/ 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
       int data;
       struct node *next;
  10 } NODE:
     // 새로운 노드를 생성하는 함수
  13 NODE *create node(int value)
  14 {
       NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
→ 17
       if(new_node == NULL)
  18
           printf("메모리 할당 오류빿");
        exit(EXIT_FAILURE);
       new_node->data = value;
       new_node->next = NULL;
  0.4
                    Edit this code
```

→ line that just executed
→ next line to execute

 Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new node data pointer to node create_node NULL (0x0) value 30 arrav pointer to NODE new_node struct NODE data pointer to node next

```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTULINATE
     기/ 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
       int data;
       struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
  13 NODE *create node(int value)
  14
       NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
\rightarrow 17
        if(new_node == NULL)
  18
           printf("메모리 할당 오류빿");
        exit(EXIT_FAILURE);
  21
       new_node->data = value;
        new_node->next = NULL;
  0.4
```

Edit this code

→ line that just executed
→ next line to execute

<< First | < Prev | Next > | Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new node data pointer to node create_node NULL (0x0) value 30 arrav pointer to NODE new_node struct NODE data pointer to node next

```
C (C17 + GNU extensions)
```

```
known limitations
      #THICTURE NOTULINATE
     기/ 연결 리스트의 노드를 나타내는 구조체
   6 typedef struct node
       int data;
       struct node *next;
  10 } NODE:
      // 새로운 노드를 생성하는 함수
  13 NODE *create node(int value)
  14 {
       NODE *new_node = (NODE *)malloc(sizeof(NODE));
  16
        if(new_node == NULL)
  18
           printf("메모리 할당 오류빿");
         exit(EXIT_FAILURE);
  21
\rightarrow 22
       new_node->data = value;
→ 23
        new_node->next = NULL;
  2.4
```

Edit this code

→ line that just executed
→ next line to execute

<< First | < Prev | Next > Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new node data pointer to node create_node NULL (0x0) value 30 arrav pointer to NODE new_node struct NODE data 30 pointer to node next

```
C (C17 + GNU extensions)
                         known limitations
     NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new_node == NULL)
     18
     19
               printf("메모리 할당 오류₩n");
               exit(EXIT_FAILURE);
           new_node->data = value;
   \rightarrow 23
           new_node->next = NULL;
     24
   → 25
           return new_node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
      OE.
                // 기치는가 비사이트 경우
                          Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                         Last >>
```

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new node data 20 pointer to node create_node NULL (0x0) value 30 arrav pointer to NODE new_node struct NODE data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
     THE C
           NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new_node == NULL)
     18
     19
               printf("메모리 할당 오류₩n");
               exit(EXIT_FAILURE);
           new_node->data = value;
           new_node->next = NULL;
     24
  → 25
           return new_node;
  → 26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
     30 {
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     OE.
               // 기치는가 비사이트 경우
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                        Last >>
```

```
Print output (drag lower right corner to resize)
                         Stack
                                          Heap
 main
                                           array
                pointer to NODE
                                           struct NODE
     my_list
                                             data
                                                   10
 insert_node
                                                   pointer to node
                                             next
              pointer to NODE*
       head
                                           arrav
      value
              30
                                           struct node
              pointer to NODE
 new node
                                             data
                                                   20
                                                   pointer to node
 create_node
                                                   NULL (0x0)
       value
               30
                                           arrav
                pointer to NODE
  new_node
                                          struct NODE
                                             data
                                                   30
                                                   pointer to node
                                             next
                                                   NULL (0x0)
```

```
C (C17 + GNU extensions)
                         known limitations
     NODE *new_node = (NODE *)malloc(sizeof(NODE));
     16
           if(new_node == NULL)
     18
               printf("메모리 할당 오류₩n");
               exit(EXIT_FAILURE);
           new_node->data = value;
           new_node->next = NULL;
     24
           return new_node;
     26
         // 연결 리스트에 노드를 추가하는 함수
         void insert_node(NODE **head, int value)
     30 {
  \rightarrow 31
           NODE *new_node = create_node(value);
   → 33
           if(*head == NULL)
     34
     OE.
               // 기치는가 비사이트 경우
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to node NULL (0x0) arrav struct NODE data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
     known limitations
```

```
JU L
     31
           NODE *new_node = create_node(value);
  \rightarrow 33
           if(*head == NULL)
              // 리스트가 비어있을 경우
     36
              *head = new node;
           else
     40
              // 리스트가 비어있지 않을 경우
              NODE *temp = *head;
               while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48
     49
            연결 리스트의 노드를 출력하는 함수
     El maid seist list(MODE resod)
                        Edit this code
ine that just executed
next line to execute
```

<< First < Prev Next > Last >> Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to NODE temp pointer to node NULL (0x0) arrav struct NODE data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
known limitations
```

```
JU L
     31
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
              // 리스트가 비어있을 경우
     36
              *head = new node;
           else
     40
              // 리스트가 비어있지 않을 경우
  \rightarrow 41
              NODE *temp = *head;
   → 42
              while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
     46
              temp->next = new_node;
     47
     48
     49
           연결 리스트의 노드를 출력하는 함수
     El maid seist list(MODE resod)
                        Edit this code
ine that just executed
```

→ line that just executed next line to execute

<< First | < Prev | Next > | Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to NODE temp pointer to node NULL (0x0) arrav struct NODE data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
     known limitations
```

```
JU L
  31
       NODE *new_node = create_node(value);
       if(*head == NULL)
  34
           // 리스트가 비어있을 경우
  36
           *head = new node;
       else
  40
           // 리스트가 비어있지 않을 경우
  41
           NODE *temp = *head;

→ 42

           while(temp->next != NULL)
  43
               temp = temp->next;
  45
  46
           temp->next = new_node;
  47
  48
  49
        연결 리스트의 노드를 출력하는 함수
  El maid seist list(MODE resod)
                    Edit this code
```

ine that just executed next line to execute

> << First < Prev Next > Last >>

Print output (drag lower right corner to resize) Stack Heap main array pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to NODE temp pointer to node NULL (0x0) arrav struct NODE data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
known limitations
```

```
JU L
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new_node;
     37
           else
     40
              // 리스트가 비어있지 않을 경우
     41
              NODE *temp = *head;
               while(temp->next != NULL)
     43
   \rightarrow 44
                  temp = temp->next;
     45
     46
               temp->next = new node;
     47
     48
     49
            연결 리스트의 노드를 출력하는 함수
     El wold print ligt (MODE shood)
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to NODE temp **⊙** pointer to node NULL (0x0) type punning: view 1 of 2 [switch views] array struct NODE data 30 pointer to node NULL (0x0)

```
C (C17 + GNU extensions)
known limitations
```

```
JU L
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new_node;
     37
           else
     40
              // 리스트가 비어있지 않을 경우
     41
              NODE *temp = *head;
   \rightarrow 42
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
              temp->next = new node;
     47
     48
     49
            연결 리스트의 노드를 출력하는 함수
     El wold print ligt (MODE shood)
                         Edit this code
ine that just executed
next line to execute
```

< Prev

Next >

Last >>

<< First

Print output (drag lower right corner to resize) Stack Heap main arrav pointer to NODE struct NODE my_list data 10 insert_node pointer to node next pointer to NODE* head arrav value 30 struct node pointer to NODE new_node data 20 pointer to NODE temp **⊙** pointer to node NULL (0x0) type punning: view 1 of 2 [switch views] array struct NODE data 30 pointer to node NULL (0x0)

```
C (C17 + GNU extensions)
known limitations
```

```
JU L
           NODE *new_node = create_node(value);
           if(*head == NULL)
     34
     35
               // 리스트가 비어있을 경우
     36
              *head = new_node;
     37
           else
               // 리스트가 비어있지 않을 경우
     41
               NODE *temp = *head;
               while(temp->next != NULL)
     44
                   temp = temp->next;
     45
   \rightarrow 46
               temp->next = new_node;
     47
   → 48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
     El moid print light (MODE Though)
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                               Next >
                                        Last >>
```

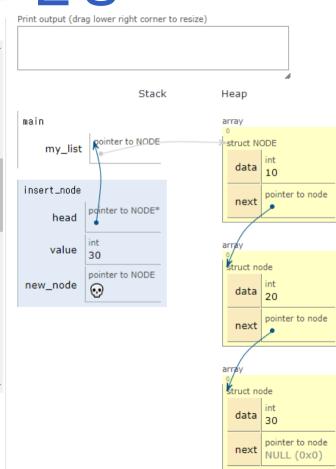
rint output (dra	g lower right corner	to resize)		
	Stack		Неар	<i>A</i>
main		Č	array 0	
my_list	pointer to NODE		struct No	ODE
my_nsc			data	int 10
insert_node	pointer to NODE*		next	pointer to node
head	. 1	Į.	$\overline{}$	
value	int 30	ě	array	
new_node	pointer to NODE		struct no	int 20
temp	pointer to NODE		next	pointer to node
			type punr [switch vi	ning: view 1 of 2 ews]
			struct no	ode
			data	int 30
			next	pointer to node

```
C (C17 + GNU extensions)
     known limitations
```

```
JU L
          NODE *new_node = create_node(value);
          if(*head == NULL)
     34
     35
              // 리스트가 비어있을 경우
     36
              *head = new_node;
     37
          else
     40
              // 리스트가 비어있지 않을 경우
     41
              NODE *temp = *head;
              while(temp->next != NULL)
     43
     44
                  temp = temp->next;
     45
     46
              temp->next = new node;
     47
  ⇒ 48 }
     49
           연결 리스트의 노드를 출력하는 함수
     El moid print light (MODE Though)
                        Edit this code
ine that just executed
```

next line to execute

<< First < Prev Next > Last >>



type punning: view 1 of 2

```
C (C17 + GNU extensions)
                        known limitations
           priniti 메모리 해제 판묘₩ /
     73
     74
     75 int main()
     76 {
           NODE *my_list = NULL;
     79
           // 연결 리스트에 노드 추가
           insert_node(&my_list, 10);
           insert_node(&my_list, 20);
   → 82
           insert_node(&my_list, 30);
     84
           // 연결 리스트 출력
           printf("연결 리스트: ");
           print_list(my_list);
     86
     87
           // 메모리 해제
           free_list(my_list);
           return O;
                         Edit this code
ine that just executed
next line to execute
                       < Prev
               << First
                               Next >
                                       Last >>
```

Print output (drag	lower right corner to	resize)		
	Stack	Н	eap	A.
main my_list	iter to NODE	0	rray struct NO	DDE
my_nst			data	int 10
			next	pointer to node
		aı	rray struct no	.do
			data	int 20
			next	pointer to node
		aı	rray	
		į	struct no	int
		-	next	pointer to node NULL (0x0)

```
C (C17 + GNU extensions)
                    known limitations
       priniti 메모리 해제 판묘₩ /
  73
  74
     int main()
  76 {
       NODE *my_list = NULL;
  79
       // 연결 리스트에 노드 추가
       insert_node(&my_list, 10);
       insert_node(&my_list, 20);
       insert_node(&my_list, 30);
  84
       // 연결 리스트 출력
       printf("연결 리스트: ");
→ 85
       print_list(my_list);
→ 86
  87
       // 메모리 해제
       free_list(my_list);
       return O;
```

Edit this code

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Print output	(drag lower right corn	er to resize)	
연결 리스	⊑:			
	Stack	ŀ	Неар	
main		ā	array	
my_list	pointer to NODE	>	struct N	DDE
my_noc			data	int 10
			next	pointer to node
		ā	array	
			Struct no	ode
			data	int 20
			next	pointer to node
			struct no	ode
			data	int 30
			next	pointer to node NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
               NODE "COMP
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
     51 void print_list(NODE *head)
   → 52 {
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
            temp = temp->next;
           printf("NULL#n");
     61
     99 // 메디지 팬테 하시
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                         Last >>
```

```
Print output (drag lower right corner to resize)
연결 리스트:
                     Stack
                                      Heap
 main
                                       arrav
           pointer to NODE
                                       struct NODE
 my_list
                                         data
                                               10
 print_list
                                                pointer to node
                                         next
            pointer to NODE
    head
           pointer to NODE
                                       arráv
   temp
                                       struct node
                                         data
                                               pointer to node
                                         next
                                      arráv
                                       struct node
                                        data 30
                                               pointer to node
                                         next
                                               NULL (0x0)
```

```
C (C17 + GNU extensions)
                         known limitations
               NODE "COMP
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
     51 void print_list(NODE *head)
  → 52 {
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
            temp = temp->next;
           printf("NULL#n");
     61
     99 // 메디지 팬테 하시
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                         Last >>
```

```
Print output (drag lower right corner to resize)
연결 리스트:
                     Stack
                                      Heap
 main
                                       arrav
           pointer to NODE
                                      struct NODE
 my_list
                                         data
                                               10
 print_list
                                                pointer to node
                                         next
           pointer to NODE
    head
           pointer to NODE
                                       arráv
    temp
                                       struct node
                                         data
                                               pointer to node
                                         next
                                      arráv
                                       struct node
                                        data 30
                                               pointer to node
                                         next
                                               NULL (0x0)
```

```
C (C17 + GNU extensions)
                         known limitations
               MODE "COMP
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
     50 // 연결 리스트의 노드를 출력하는 함수
     51 void print_list(NODE *head)
     52
   → 53
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
            temp = temp->next;
           printf("NULL\n");
     60
     61
     99 // 메디지 팬테 하시
                         Edit this code
ine that just executed
next line to execute
                << First
                       < Prev
                                Next >
                                        Last >>
```

Print output (drag lower right corner to resize) 연결 리스트:				
	Stack	ı	Неар	<i>M</i>
main			array 0	
my_list	pointer to NODE		struct N	DDE
/-			data	int 10
print_lis	pointer to MODE		next	pointer to node
head	• / / / / / / / / / / / / / / / / / / /		_	<u> </u>
temp	pointer to NODE		array	
			struct no	ode
			data	int 20
			next	pointer to node
		,	array	
			struct no	ode
			data	int 30
			next	pointer to node NULL (0x0)

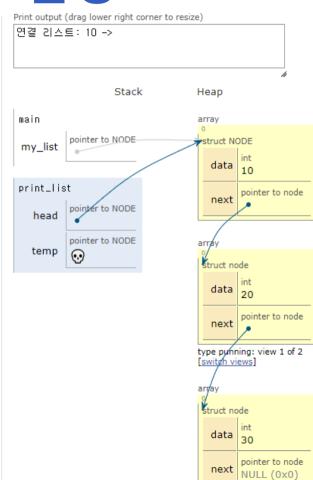
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
     52
           NODE *temp = head;
   \rightarrow 54
           while(temp != NULL)
               printf("%d -> ", temp->data);
     57
            temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```

Print output (drag lower right corner to resize) 연결 리스트: Stack Heap main arrav pointer to NODE struct NODE my_list data 10 print_list pointer to node next pointer to MODE head pointer to NODE arráv temp struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

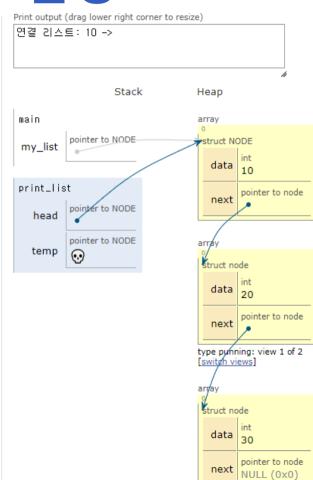
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                    temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
   \rightarrow 56
               printf("%d -> ", temp->data);
   → 57
               temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                         Last >>
                                  Next >
```

```
Print output (drag lower right corner to resize)
연결 리스트: 10 ->
                     Stack
                                      Heap
 main
                                       arrav
           pointer to NODE
                                      struct NODE
 my_list
                                         data
                                               10
 print_list
                                               pointer to node
                                         next
           pointer to MODE
   head
           pointer to NODE
                                       arráv
   temp
                                       struct node
                                         data
                                               pointer to node
                                         next
                                      arráv
                                       struct node
                                        data 30
                                               pointer to node
                                         next
                                               NULL (0x0)
```

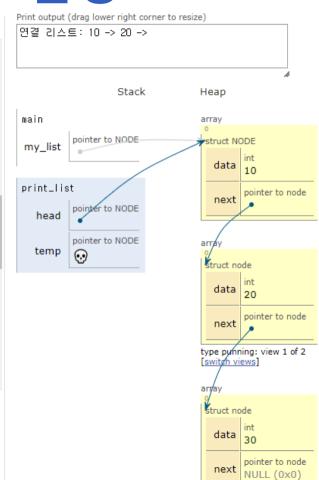
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                    temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
   \rightarrow 57
            temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```



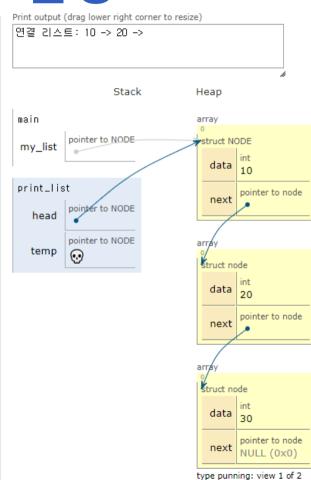
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
      42
                while(temp->next != NULL)
      43
      44
                    temp = temp->next;
      45
      46
               temp->next = new_node;
      47
      48 }
      49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
      52
           NODE *temp = head;
   \rightarrow 54
            while(temp != NULL)
               printf("%d -> ", temp->data);
      57
               temp = temp->next;
           printf("NULL#n");
      60
      61
      99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```



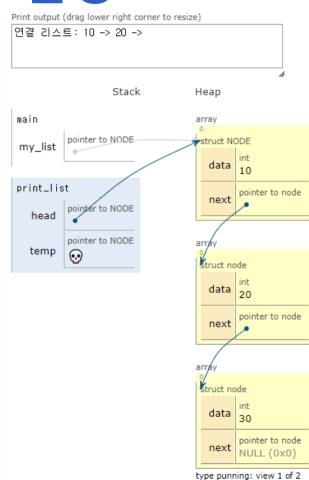
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                    temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
   \rightarrow 56
               printf("%d -> ", temp->data);
   → 57
               temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                          Last >>
                                 Next >
```



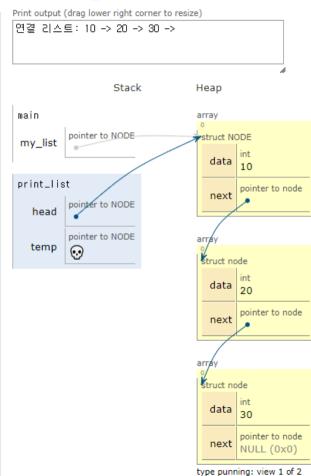
```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                    temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
   \rightarrow 57
            temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```



```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
                while(temp->next != NULL)
      43
      44
                    temp = temp->next;
      45
      46
               temp->next = new_node;
      47
      48 }
      49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
      52
           NODE *temp = head;
   \rightarrow 54
            while(temp != NULL)
               printf("%d -> ", temp->data);
      57
               temp = temp->next;
           printf("NULL#n");
      60
      61
      99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```



```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
      42
                while(temp->next != NULL)
      43
      44
                    temp = temp->next;
      45
      46
               temp->next = new_node;
      47
      48 }
      49
         // 연결 리스트의 노드를 출력하는 함수
         void print_list(NODE *head)
      52
           NODE *temp = head;
            while(temp != NULL)
      55
   \rightarrow 56
               printf("%d -> ", temp->data);
   → 57
               temp = temp->next;
            printf("NULL\n");
      60
      61
      99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```



```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
   \rightarrow 57
            temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> Stack Heap main arrav pointer to NODE struct NODE my_list data 10 print_list pointer to node next pointer to NODE head pointer to NODE arráv temp NULL (0x0) struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                          known limitations
                NODE "COMP
     42
                while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print list(NODE *head)
     52
           NODE *temp = head;
   \rightarrow 54
           while(temp != NULL)
               printf("%d -> ", temp->data);
     57
            temp = temp->next;
           printf("NULL#n");
     60
     61
     99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                         < Prev
                                 Next >
                                          Last >>
```

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> Stack Heap main arrav pointer to NODE struct NODE my_list data 10 print_list pointer to node next pointer to NODE head pointer to NODE arráv temp NULL (0x0) struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
               NODE "COMP
     42
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
     55
               printf("%d -> ", temp->data);
            temp = temp->next;
  → 59
           printf("NULL#n");
  → 60 }
     61
     99 // 메디지 팬테 하시
                         Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                         Last >>
```

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 print_list pointer to node next pointer to NODE head pointer to NODE arráv temp NULL (0x0) struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
               NODE "COMP
     42
               while(temp->next != NULL)
     43
     44
                   temp = temp->next;
     45
     46
               temp->next = new_node;
     47
     48 }
     49
         // 연결 리스트의 노드를 출력하는 함수
         void print list(NODE *head)
     52
           NODE *temp = head;
           while(temp != NULL)
               printf("%d -> ", temp->data);
               temp = temp->next;
           printf("NULL#n");
  ⇒ 60 }
     61
      99 // 메디지 팬테 하시
                          Edit this code
ine that just executed
next line to execute
                << First
                        < Prev
                                Next >
                                         Last >>
```

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 print_list pointer to node next pointer to NODE head pointer to NODE arráv temp NULL (0x0) struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
```

```
known limitations
       하내다 메모리 해제 판표째 가
  74
     int main()
  76
       NODE *my_list = NULL;
  79
       // 연결 리스트에 노드 추가
       insert_node(&my_list, 10);
       insert_node(&my_list, 20);
       insert_node(&my_list, 30);
  84
       // 연결 리스트 출력
       printf("연결 리스트: ");
       print_list(my_list);
→ 86
  87
       // 메모리 해제
       free_list(my_list);
       return O;
```

Edit this code

→ line that just executed
→ next line to execute

<< First | < Prev | Next > Last >>

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main array pointer to NODE struct NODE my_list data 10 pointer to node next arráv struct node data pointer to node next arráv struct node data pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
```

```
known limitations
     NODE "COMP HOUGE
     while(temp != NULL)
55
56
         printf("%d -> ", temp->data);
        temp = temp->next;
58
     printf("NULL#n");
61
   // 메모리 해제 함수
63 void free list(NODE *head)
     NODE *temp;
     while(head != NULL)
67
      temp = head;
        head = head->next;
     free(temp);
     printf("메모리 해제 완료₩n");
7.4
```

Edit this code

→ line that just executed→ next line to execute



Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main array pointer to NODE struct NODE my_list data 10 free_list pointer to node next pointer to NODE head pointer to NODE arráv temp struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
           NODE "COMP HOUGE
           while(temp != NULL)
     55
     56
               printf("%d -> ", temp->data);
               temp = temp->next;
     58
           printf("NULL#n");
     61
         // 메모리 해제 함수
     63 void free list(NODE *head)
  → 64 {
           NODE *temp;
           while(head != NULL)
     67
           temp = head;
               head = head->next;
           free(temp);
           printf("메모리 해제 완료₩n");
     7.4
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 free_list pointer to node next pointer to NODE head pointer to NODE arráv temp struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                          known limitations
            NODE "COMP HOUGE
           while(temp != NULL)
     55
     56
               printf("%d -> ", temp->data);
               temp = temp->next;
     58
           printf("NULL#n");
     61
         // 메모리 해제 함수
         void free list(NODE *head)
     64
           NODE *temp;
           while(head != NULL)
   \rightarrow 66
     67
   → 68
           temp = head;
               head = head->next;
            free(temp);
           printf("메모리 해제 완료₩n");
     7.4
                          Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 free_list pointer to node next pointer to NODE head pointer to NODE arráv temp struct node data pointer to node next arráv struct node data 30 pointer to node next NULL (0x0)

```
C (C17 + GNU extensions)
                          known limitations
           NODE "COMP HOUGE
           while(temp != NULL)
     55
     56
               printf("%d -> ", temp->data);
               temp = temp->next;
     58
           printf("NULL#n");
     61
         // 메모리 해제 함수
         void free list(NODE *head)
     64
           NODE *temp;
           while(head != NULL)
     67
   \rightarrow 68
           temp = head;
   → 69
           head = head->next;
           free(temp);
     71
           printf("메모리 해제 완료₩n");
     7.4
                          Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 free_list pointer to node next pointer to MODE head pointer to NODE arráv temp struct node data pointer to node next arráv struct node data 30 pointer to node

next

NULL (0x0)

```
C (C17 + GNU extensions)
                         known limitations
           NODE "COMP HOUGE
           while(temp != NULL)
     55
     56
               printf("%d -> ", temp->data);
               temp = temp->next;
     58
           printf("NULL#n");
     61
         // 메모리 해제 함수
         void free list(NODE *head)
     64
           NODE *temp;
           while(head != NULL)
     67
            temp = head;
  → 69
              head = head->next;
   → 70
           free(temp);
     71
           printf("메모리 해제 완료₩n");
     7.4
                         Edit this code
ine that just executed
next line to execute
```

<< First

< Prev

Next >

Last >>

Print output (drag lower right corner to resize) 연결 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main arrav pointer to NODE struct NODE my_list data 10 free_list pointer to node next pointer to MODE head pointer to NODE temp array struct NODE data pointer to node next type punning: view 1 of 2 [switch views] arday struct node data 30 pointer to node next | NULL (0x0)

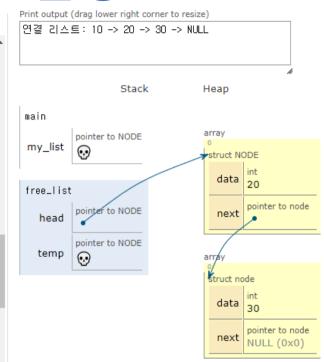
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLONE
  54
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
  57
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
  65
        NODE *temp;
        while(head != NULL)
  67
  68
            temp = head;
  69
            head = head->next;
\rightarrow 70
            free(temp);
  71
        printf("메모리 해제 완료₩n");
  73
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: • means a pointer points to memory that is either unallocated or misaligned with data boundaries. • locations are approximate and may not match the pointer's real address.

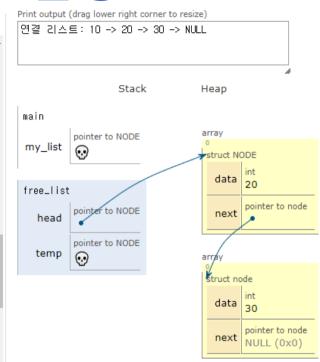
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLONE
  54
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
  57
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
  65
        NODE *temp;
        while(head != NULL)
\rightarrow 66
  67
→ 68
            temp = head;
            head = head->next;
            free(temp);
  71
        printf("메모리 해제 완료₩n");
  73
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: • means a pointer points to memory that is either unallocated or misaligned with data boundaries. • locations are approximate and may not match the pointer's real address.

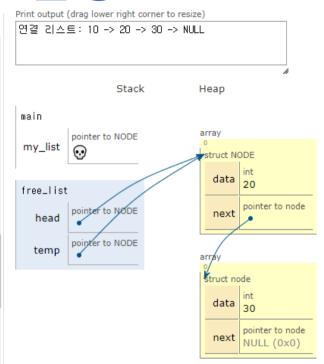
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLONE
  54
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
  57
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
  65
        NODE *temp;
        while(head != NULL)
  67
→ 68
            temp = head;
→ 69
            head = head->next;
            free(temp);
  71
        printf("메모리 해제 완료₩n");
  73
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. locations are approximate and may not match the pointer's real address.

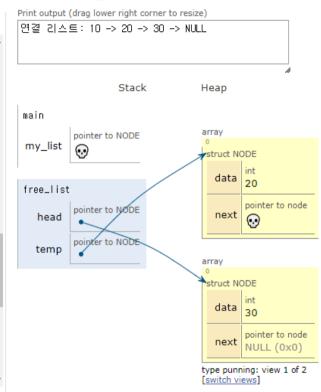
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLONE
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
  68
           temp = head;
→ 69
           head = head->next;
→ 70
            free(temp);
        printf("메모리 해제 완료₩n");
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: means a pointer points to memory that is either

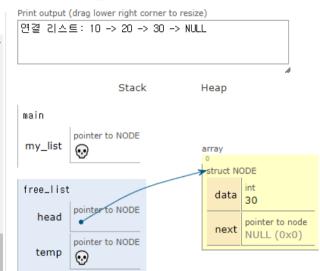
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLOND
        while(temp != NULL)
  55
  56
            printf("%d -> ". temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
           temp = head;
  69
           head = head->next;
→ 70.
           free(temp);
        printf("메모리 해제 완료₩n");
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

```
C/C++ details: none [default view]
```

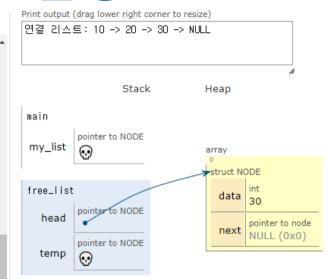
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLOND
        while(temp != NULL)
  55
  56
            printf("%d -> ". temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
  65
        NODE *temp;
        while(head != NULL)
\rightarrow 66
  67
→ 68
            temp = head;
            head = head->next;
            free(temp);
        printf("메모리 해제 완료₩n");
  7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

```
C/C++ details: none [default view]
```

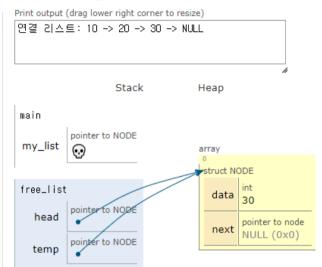
```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLOND
        while(temp != NULL)
  55
            printf("%d -> ". temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
\rightarrow 68
            temp = head;
→ 69
            head = head->next;
            free(temp);
        printf("메모리 해제 완료₩n");
   7.4
```

Edit this code

→ line that just executed
→ next line to execute





Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

```
C/C++ details: none [default view]
```

```
C (C17 + GNU extensions)
```

```
known limitations
        NODE PLOND
        while(temp != NULL)
  55
            printf("%d -> ". temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
            temp = head;
\rightarrow 69
            head = head->next;
→ 70
            free(temp);
        printf("메모리 해제 완료₩n");
  7.4
```

Edit this code

→ line that just executed
→ next line to execute



Print output (drag lower right corner to resize) 연경 리스트: 10 -> 20 -> 30 -> NULL Stack Heap main pointer to NODE my_list array struct NODE free_list data pointer to NODE head NULL (Ox pointer to node next NULL (0x0) pointer to NODE temp

Note: Pointers to structs, unions, and C++ objects may be ambiguous since a struct/object's address is the same as its first member and a union's address is the same as all its members. Select 'show memory addresses' below to see more details:

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
        NODE "COMP.
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
            temp = head;
  69
            head = head->next;
\rightarrow 70
            free(temp);
  71
        printf("메모리 해제 완료₩n");
  7.4
```

Edit this code

→ line that just executed→ next line to execute



Print output (drag lower right corner to resize)
연결 리스트: 10 -> 20 -> 30 -> NULL

Stack Heap

main

my_list pointer to NODE

free_list

head pointer to NODE

NULL (0x0)

pointer to NODE

we head pointer to NODE

NULL (0x0)

pointer to NODE

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
     NODE "COMP.
     while(temp != NULL)
55
56
         printf("%d -> ", temp->data);
         temp = temp->next;
58
     printf("NULL\n");
60
61
   // 메모리 해제 함수
   void free list(NODE *head)
64
     NODE *temp;
     while(head != NULL)
67
         temp = head;
         head = head->next;
         free(temp);
     printf("메모리 해제 완료₩n");
7.4
```

Edit this code

→ line that just executed→ next line to execute



Print output (drag lower right corner to resize)
연결 리스트: 10 -> 20 -> 30 -> NULL

Stack Heap

main

my_list

pointer to NODE

NULL (0x0)

pointer to NODE

NULL (0x0)

pointer to NODE

NULL (0x0)

Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
        NODE "COMP.
        while(temp != NULL)
  55
  56
            printf("%d -> ", temp->data);
            temp = temp->next;
  58
        printf("NULL\n");
  60
  61
      // 메모리 해제 함수
      void free list(NODE *head)
  64
        NODE *temp;
        while(head != NULL)
  67
           temp = head;
            head = head->next;
           free(temp);
        printf("메모리 해제 완료₩n");
→ 73
```

Edit this code

→ line that just executed
 → next line to execute

7.4



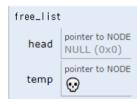
Print output (drag lower right corner to resize)

연결 리스트: 10 -> 20 -> 30 -> NULL 메모리 해제 완료

Stack Heap

main

my_list pointer to NODE



Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
        NODE "COMP.
        while(temp != NULL)
   55
   56
            printf("%d -> ", temp->data);
            temp = temp->next;
   58
        printf("NULL\n");
   60
   61
      // 메모리 해제 함수
      void free list(NODE *head)
   64
        NODE *temp;
        while(head != NULL)
   67
            temp = head;
            head = head->next;
            free(temp);
        printf("메모리 해제 완료₩n");
= 73
```

Edit this code

→ line that just executed
 → next line to execute

7.4



Print output (drag lower right corner to resize)

연결 리스트: 10 -> 20 -> 30 -> NULL 메모리 해제 완료

Stack Heap

main

my_list pointer to NODE

Note: means a pointer points to memory that is either unallocated or misaligned with data boundaries. locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
       마마디 메모리 에게 전표째 사
  74
     int main()
  76
       NODE *mv list = NULL;
  79
       // 연결 리스트에 노드 추가
       insert_node(&my_list, 10);
       insert_node(&my_list, 20);
       insert_node(&my_list, 30);
  84
       // 연결 리스트 출력
       printf("연결 리스트: ");
       print_list(my_list);
  86
  87
       // 메모리 해제
→ 89
       free_list(my_list);
→ 91
       return O;
```

Edit this code

→ line that just executed→ next line to execute

<< First | < Prev | Next > Last >>

Print output (drag lower right corner to resize)

연결 리스트: 10 -> 20 -> 30 -> NULL 메모리 해제 완료

Stack Heap

main

my_list pointer to NODE

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C (C17 + GNU extensions)

```
known limitations
        마마디 메모리 에게 전표째 사
  74
      int main()
  76
        NODE *mv list = NULL;
  79
        // 연결 리스트에 노드 추가
        insert_node(&my_list, 10);
        insert_node(&my_list, 20);
        insert_node(&my_list, 30);
  84
        // 연결 리스트 출력
        printf("연결 리스트: ");
        print_list(my_list);
  86
  87
        // 메모리 해제
        free_list(my_list);
\rightarrow 91
        return O;
92
```

Edit this code

→ line that just executed → next line to execute

Print output (drag lower right corner to resize)

연결 리스트: 10 -> 20 -> 30 -> NULL 메모리 해제 완료

Stack Heap

main

my_list pointer to NODE

Note: •• means a pointer points to memory that is either unallocated or misaligned with data boundaries. •• locations are approximate and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

