

PRACTICAL – 04

Q. Write a program for creation of Linked list.

Program:

```
// create a link list
#include<stdio.h>
#include<malloc.h>

struct node
{
};

int data;

struct node * next;

void main()
{
    struct node *p, *q, *r, *s;

    p=(struct node*)malloc(sizeof(struct node));
    p-> data = 5;
    p-> next = q;

    q=(struct node*)malloc(sizeof(struct node));
    q-> data = 10;
    q-> next = r;

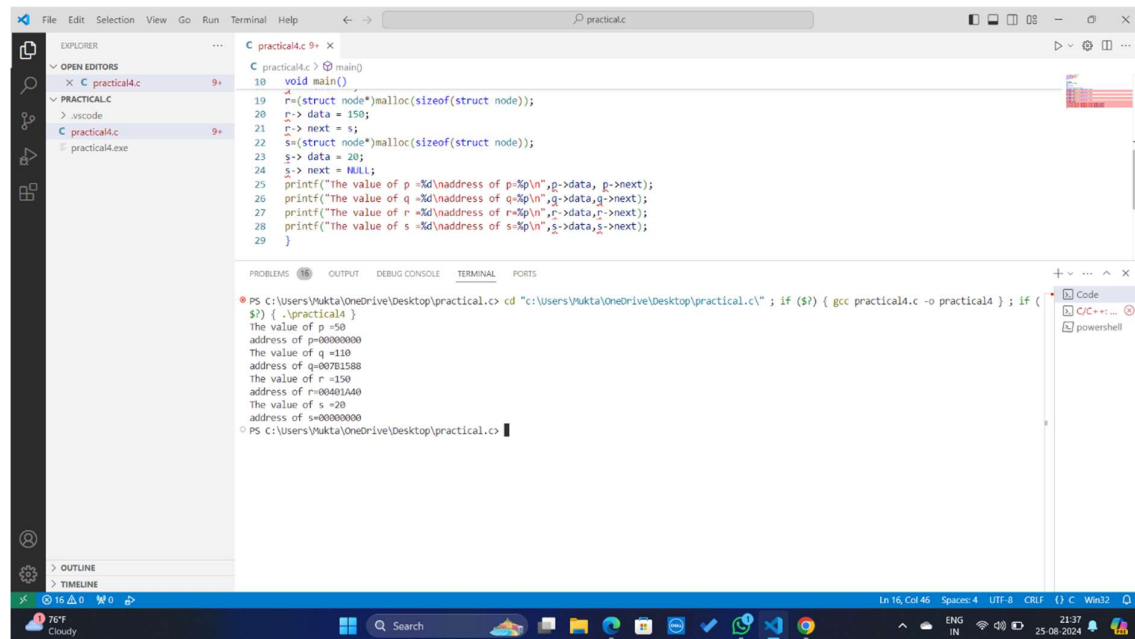
    r=(struct node*)malloc(sizeof(struct node));
    r-> data = 15;
    r-> next = s;

    s=(struct node*)malloc(sizeof(struct node));
    s-> data = 20;
    s-> next = NULL;

    printf("The value of p =%d\naddress of p=%p\n",p->data, p->next);
    printf("The value of q =%d\naddress of q=%p\n",q->data,q->next);
    printf("The value of r =%d\naddress of r=%p\n",r->data,r->next);
```

```
printf("The value of s =%d\naddress of s=%p\n",s->data,s->next);
}
```

Output:



The screenshot shows a Visual Studio Code window with a C file named 'practical4.c'. The code in the editor is as follows:

```
10 void main()
11 {
12     p=(struct node*)malloc(sizeof(struct node));
13     p->data = 150;
14     p->next = s;
15     s=(struct node*)malloc(sizeof(struct node));
16     s->data = 20;
17     s->next = NULL;
18     printf("The value of p =%d\naddress of p=%p\n",p->data, p->next);
19     printf("The value of q =%d\naddress of q=%p\n",q->data,q->next);
20     printf("The value of r =%d\naddress of r=%p\n",r->data,r->next);
21     printf("The value of s =%d\naddress of s=%p\n",s->data,s->next);
22 }
```

The terminal output shows the execution of the program:

```
PS C:\Users\Nukta\OneDrive\Desktop\practical.c> cd "C:\Users\Nukta\OneDrive\Desktop\practical.c" ; if ($?) { gcc practical4.c -o practical4 } ; if ($?) { .\practical4 }
The value of p =50
address of p =00000000
The value of q =110
address of q =007B1588
The value of r =150
address of r =00401A40
The value of s =20
address of s =00000000
PS C:\Users\Nukta\OneDrive\Desktop\practical.c>
```

Q. Write a program to insert an element at the beginning of Linked list.

Program:

```
// Insert an element at beginning of Linked list
```

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

```
#include<malloc.h>
```

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

```
struct node
```

```
{
```

```
int data;
```

```
struct node*next;
```

```
};struct node *s;
```

```
insert_beg(struct node**S)
```

```
{
```

```
struct node*p;
```

```

p=(struct node*)malloc(sizeof(struct node));
printf("Enter new node data at begining\n");
scanf("%d",&p->data);
p->next=s;
s=p;
}
void main()
{
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getch();
} while (ch=='y');
p->next=NULL;
insert_beg(&s);
printf("list of data are\n");
while(s!=NULL)
{
printf("%d\n",s->data);
s=s->next;

```

```

}

getch();

}

```

Output:

The screenshot shows a Visual Studio Code editor with a C program named 'practical4.c' open. The program implements a linked list with a 'next' pointer. The terminal output shows the program's execution, including prompts for entering data and the resulting list.

```

C:\practical4.c 9+ X
C:\practical4.c 9+ X
18 }
19 void main()
20 {
21     struct node *p,*q;
22     char ch;
23     p=(struct node*)malloc(sizeof(struct node));
24     printf("Enter first node data\n");
25     scanf("%d",&p->data);
26     s=p;
27     do
28     {
29         q=(struct node*)malloc(sizeof(struct node));
30         printf("Enter value of next node\n");
31         scanf("%d",&q->data);
32         p->next=q;
33         p=q;
34         printf("press Y for add new node \n");
35         ch=getch();
36     } while (ch=='Y');
37     p->next=NULL;
38     insert_beg(&s);
39     printf("list of data are\n");
40     while(s!=NULL)
41     {
42         printf("%d ",s->data);
43         s=s->next;
44     }
45     printf("\n");
46 }

```

Terminal Output:

```

Enter first node data
5
Enter value of next node
6
press Y for add new node
Enter new node data at beginning
Y
list of data are
5 6

```

Q. Write a Program to insert an element at a specific position of linked list.

Program:

```
// insert element at specific position of linked list
```

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

```
#include<malloc.h>
```

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

```
struct node
```

```
{
```

```
int data;
```

```
struct node*next;
```

```

};struct node *s;
insert_after(struct node**S)
{
int key;
struct node*p,*q,*first;
printf("Enter element after which want to insert\n");
scanf("%d",&key);
first=s;
while(s->data!=key)
{
}
s=s->next;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter new node data at any position\n");
scanf("%d",&p->data);
if(s->data==key)
{
q=s->next;
s->next=p;
p->next=q;
s=first;
}
else{
printf("not found\n");
}
}

void main()
{

```

```

struct node *p,*q;
char ch;

p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
    q=(struct node*)malloc(sizeof(struct node));
    printf("Enter value of next node\n");
    scanf("%d",&q->data);
    p->next=q;
    p=q;
    printf("press Y for add new node \n");
    ch=getch();
} while (ch=='y');
p->next=NULL;
insert_after(&s);
printf("list of data are:\n");
while(s!=NULL)
{
}
printf("%d\n",s->data);
s=s->next;
getch();
}

```

Output:

```
40 // Function to print the linked list
41 void print_list(struct node* head) {
42     struct node* temp = head;
43     while (temp != NULL) {
44         printf("%d\n", temp->data);
45         temp = temp->next;
46     }
47 }
48
49 int main() {
50     struct node *head = NULL, *p, *q;
51     char ch;
52
53     // Create the linked list
54     do {
55         p = (struct node*)malloc(sizeof(struct node));
56         printf("Enter node data: ");
57         scanf("%d", &p->data);
58     } while (ch != 'y');
```

Enter node data: 5
Press Y to add another node: y
Enter node data: 5
Press Y to add another node: y
Enter node data: 6
Press Y to add another node:
Enter element after which you want to insert: 5
Enter new node data: 3

List of data:
5
3
6

Q. Write a Program to insert an element at end of the linked list.

Program:

```
// insert at end of linked list
```

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

```
#include<malloc.h>
```

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

```
struct node
```

```
{
```

```
int data;
```

```
struct node*next;
```

```
};struct node *s;
```

```
void insert_end(struct node**S)
```

```
{
```

```
struct node*t,*first;
```

```
t=(struct node*)malloc(sizeof(struct node));
```

```
printf("Enter new node data at End\n");
```

```

scanf("%d",&t->data);
first=s;
while(s->next!=NULL)
{
}
s=s->next;
s->next=t;
t->next=NULL;
s=first;
}
void main()
{
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getchar();
ch=getchar();
} while (ch=='y');

```



```

p->next=NULL;

insert_end(&s);

printf("list of data are\n");

while(s!=NULL)

{

printf("%d\n",s->data);

s=s->next;

}

getchar();

}

```

Output:

The screenshot shows a Visual Studio Code editor with a C program in a file named 'practical.c'. The program implements a linked list with functions for inserting at the end, printing the list, and freeing memory. The terminal output shows the execution of the program, where the user enters data for four nodes (5, 6, 8, 4) and the program prints the list of data as 5, 6, 8, 4.

```

C 4.c > _
38 int main() {
62     insert_end(&head);
63
64     // Print the linked list
65     printf("List of data:\n");
66     print_list(head);
67
68     // Free the allocated memory
69     struct node* temp;
70     while (head != NULL) {
71         temp = head;
72         head = head->next;
73         free(temp);
74     }
75     return 0;
76
77 }
78

```

```

PS C:\Users\Vukta\OneDrive\Desktop\practical.c> cd "C:\Users\Vukta\OneDrive\Desktop\practical.c\" ; if ($?) { gcc 4.c -o 4 } ; if ($?) { .\4 }
Enter node data: 5
Press Y to add another node: y
Enter node data: 6
Press Y to add another node: y
Enter node data: 8
Press Y to add another node: y
Enter node data: 4
List of data:
5
6
8
4
PS C:\Users\Vukta\OneDrive\Desktop\practical.c>

```

Q . Write a program to delete an element at beginning of linked list.

Program:

```
// delete from beg
```

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

```
#include<malloc.h>
```

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node*next;
```

```
};struct node *s;
```

```
void delete_beg(struct node**S)
```

```
{
```

```
    struct node*p;
```

```
if(s==NULL)
```

```
{
```

```
    printf("Under flow");
```

```
    return ;
```

```
}
```

```
    p=s;
```

```
    s=p->next;
```

```
free(p);
```

```
}
```

```
void main()
```

```
{
```

```
struct node *p,*q;
```

```
char ch;
```

```
p=(struct node*)malloc(sizeof(struct node));
```

```
printf("Enter first node data\n");
```

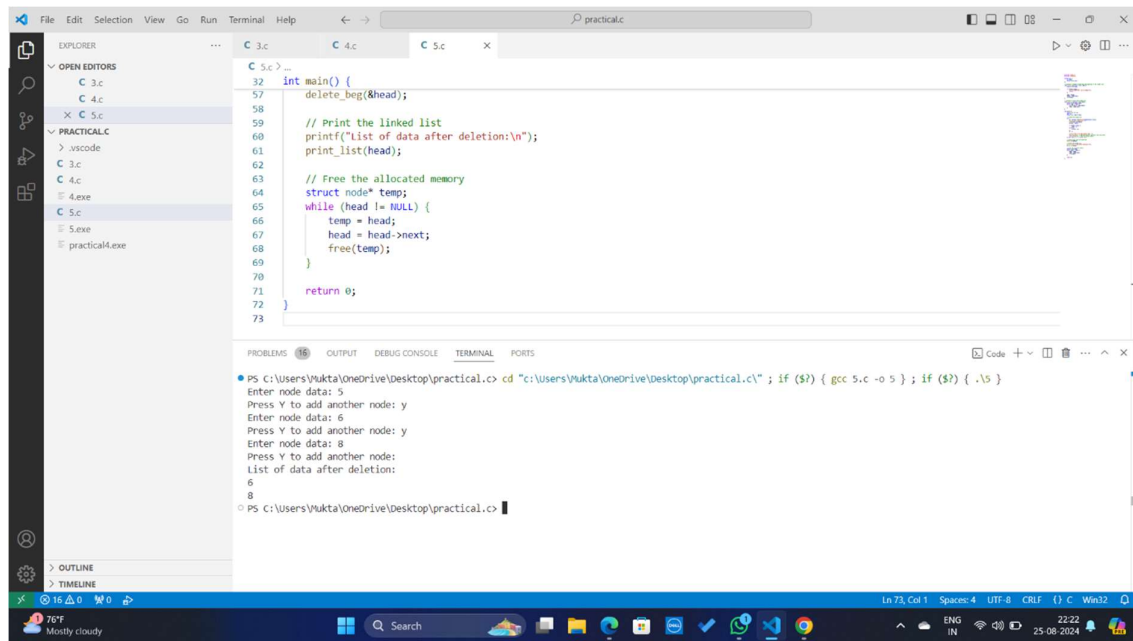
```
scanf("%d",&p->data);
```

```

s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getch();
} while (ch=='y');
p->next=NULL;
delete_beg(&s);
printf("list of data are\n");
while(s!=NULL)
{
}
printf("%d\n",s->data);
s=s->next;
getch();
}

```

Output:



Q. Write a program delete specific element from linked list.

Program:

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

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

```

struct node {
    int data;
    struct node* next;
};

```

```
// Function to delete a specific node from the linked list
```

```
void delete_specific(struct node** head) {
```

```
    int key;
```

```
    struct node *temp, *prev;
```

```
    if (*head == NULL) {
```

```

    printf("List is empty.\n");
    return;
}

printf("Enter element to delete: ");
scanf("%d", &key);

// If the node to be deleted is the head node
if ((*head)->data == key) {
    temp = *head;
    *head = temp->next;
    free(temp);
    return;
}

// Traverse the list to find the node to delete
prev = *head;
temp = (*head)->next;

while (temp != NULL && temp->data != key) {
    prev = temp;
    temp = temp->next;
}

// Node with the key not found
if (temp == NULL) {
    printf("Element %d not found in the list.\n", key);
    return;
}

```

```
// Remove the node

prev->next = temp->next;

free(temp);

}
```

```
// Function to print the linked list

void print_list(struct node* head) {

    struct node* temp = head;

    while (temp != NULL) {

        printf("%d\n", temp->data);

        temp = temp->next;

    }

}
```

```
int main() {

    struct node *p, *q;

    char ch;

    struct node *head = NULL;

    // Create the linked list

    do {

        p = (struct node*)malloc(sizeof(struct node));

        printf("Enter node data: ");

        scanf("%d", &p->data);

        p->next = NULL;

        if (head == NULL) {

            head = p;

        }

    } while (ch != 'q');
```

```

    } else {
        q->next = p;
    }
    q = p;

    printf("Press Y to add another node: ");
    ch = getchar(); // Read the newline character from the buffer
    ch = getchar(); // Read the actual choice
} while (ch == 'Y' || ch == 'y');

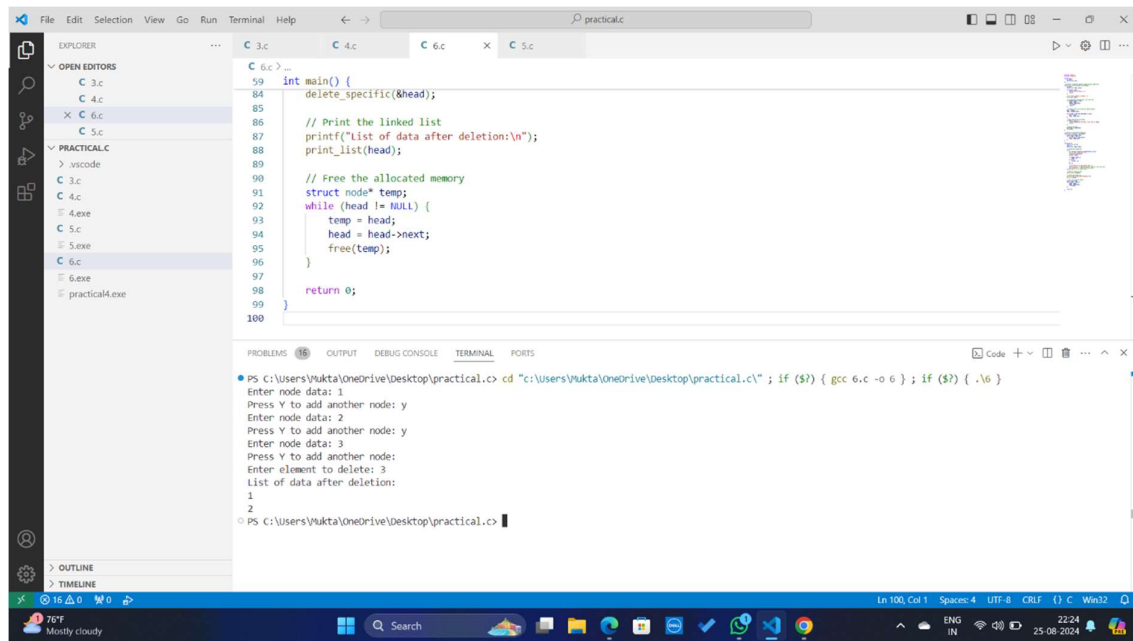
// Delete a specific node
delete_specific(&head);

// Print the linked list
printf("List of data after deletion:\n");
print_list(head);

// Free the allocated memory
struct node* temp;
while (head != NULL) {
    temp = head;
    head = head->next;
    free(temp);
}

return 0;
}Output:

```



Q. Write a program to delete an element at end of linked list.

Program:

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

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

```
struct node {
```

```
    int data;
```

```
    struct node* next;
```

```
};
```

```
// Function to delete the last node from the linked list
```

```
void delete_end(struct node** head) {
```

```
    struct node *temp, *prev;
```

```
    if (*head == NULL) {
```

```
        printf("List is empty.\n");
```

```
        return;
```



```
}
```

```
if ((*head)->next == NULL) {  
    // Only one node in the list  
    free(*head);  
    *head = NULL;  
    return;  
}
```

```
temp = *head;  
while (temp->next != NULL) {  
    prev = temp;  
    temp = temp->next;  
}
```

```
// Now temp points to the last node  
prev->next = NULL;  
free(temp);  
}
```

```
// Function to print the linked list  
void print_list(struct node* head) {  
    struct node* temp = head;  
    while (temp != NULL) {  
        printf("%d\n", temp->data);  
        temp = temp->next;  
    }  
}
```

```

int main() {
    struct node *p, *q;

    char ch;

    struct node *head = NULL;

    // Create the linked list
    do {
        p = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &p->data);
        p->next = NULL;

        if (head == NULL) {
            head = p;
        } else {
            q->next = p;
        }
        q = p;

        printf("Press Y to add another node: ");
        ch = getchar(); // Read the newline character from the buffer
        ch = getchar(); // Read the actual choice
    } while (ch == 'Y' || ch == 'y');

    // Delete the last node
    delete_end(&head);

    // Print the linked list
    printf("List of data after deletion:\n");

```

```
print_list(head);
```

```
// Free the allocated memory
```

```
struct node* temp;
```

```
while (head != NULL) {
```

```
    temp = head;
```

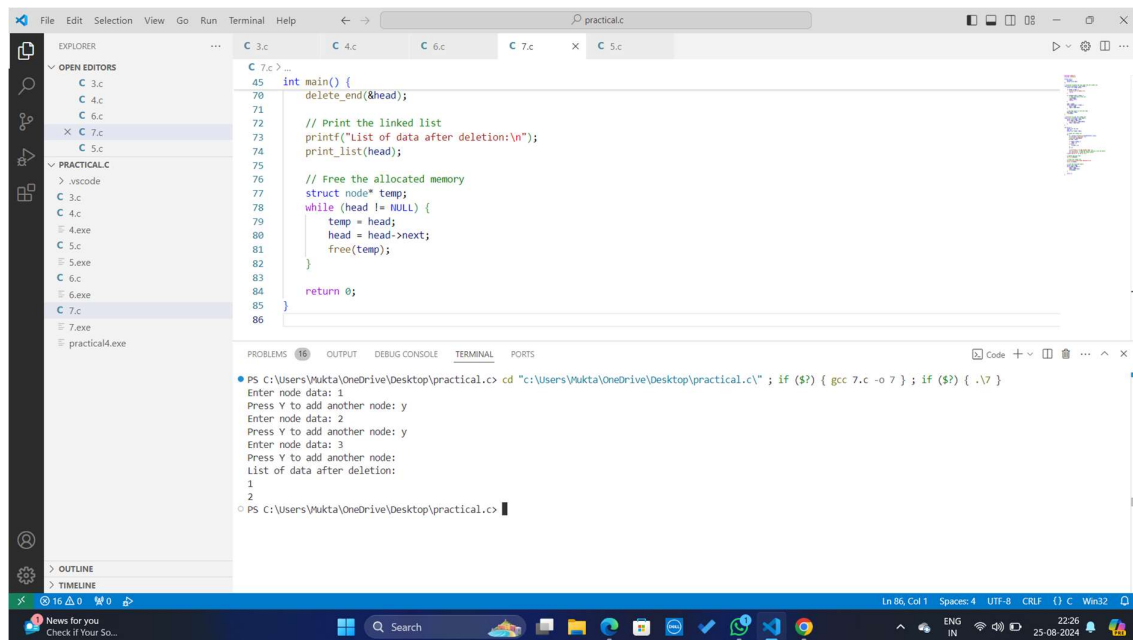
```
    head = head->next;
```

```
    free(temp);
```

```
}
```

```
return 0;
```

}Output:



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'practical.c' with files 3.c, 4.c, 5.c, 6.c, 7.c, and 7.exe. The main editor window displays the source code for 'practical.c', which includes a linked list deletion function. The output panel at the bottom shows the execution of the program, where the user enters node data (1, 2, 3) and the program prints the list after deletion (1, 2).

```
45 int main() {
70     delete_end(&head);
71
72     // Print the linked list
73     printf("List of data after deletion:\n");
74     print_list(head);
75
76     // Free the allocated memory
77     struct node* temp;
78     while (head != NULL) {
79         temp = head;
80         head = head->next;
81         free(temp);
82     }
83
84     return 0;
85 }
86
```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nukta\OneDrive\Desktop\practical.c> cd "C:\Users\Nukta\OneDrive\Desktop\practical.c\" ; if ($?) { gcc 7.c -o 7 } ; if ($?) { .\7 }
Enter node data: 1
Press Y to add another node: y
Enter node data: 2
Press Y to add another node: y
Enter node data: 3
Press Y to add another node:
List of data after deletion:
1
2
PS C:\Users\Nukta\OneDrive\Desktop\practical.c>
```