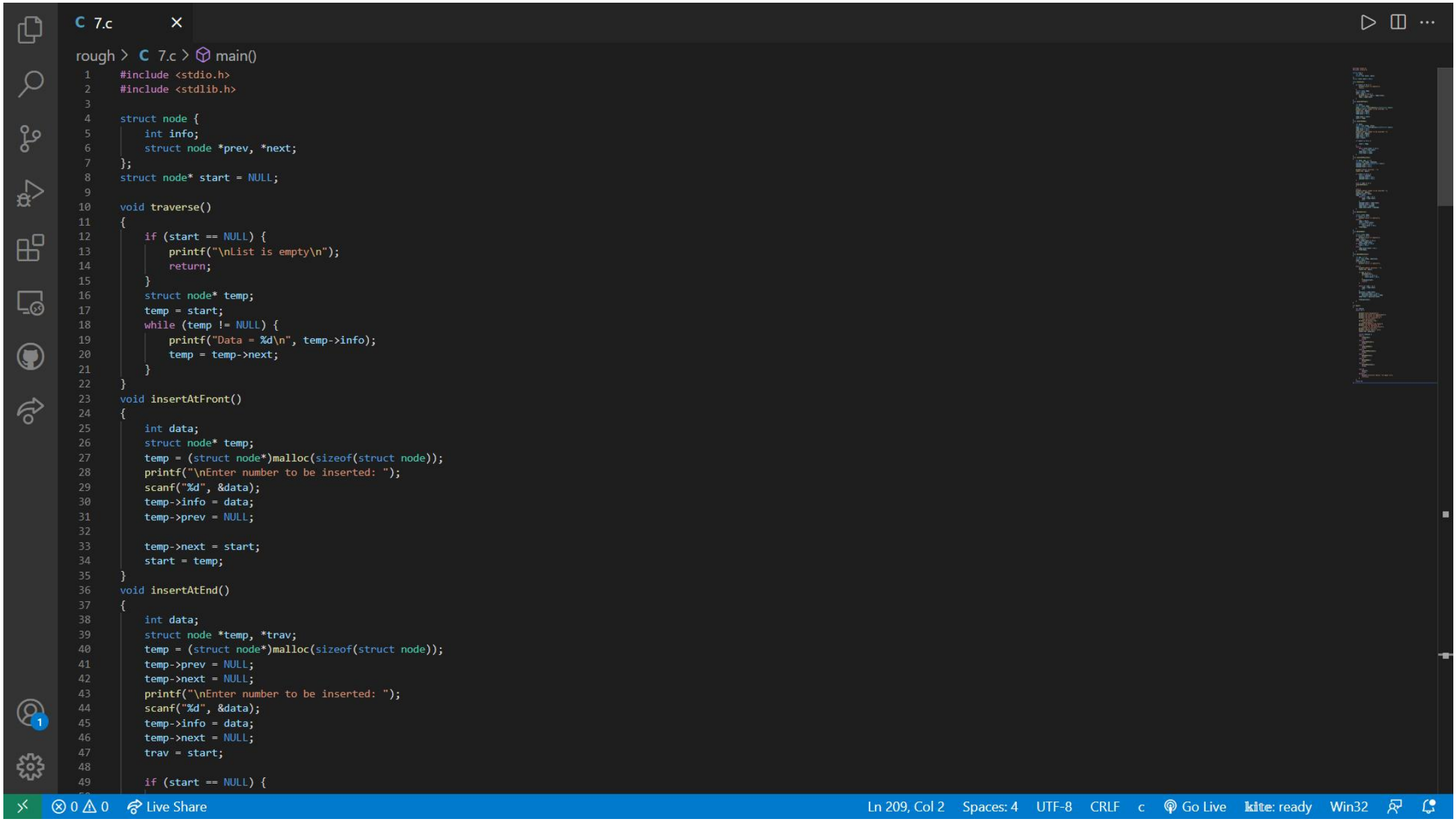
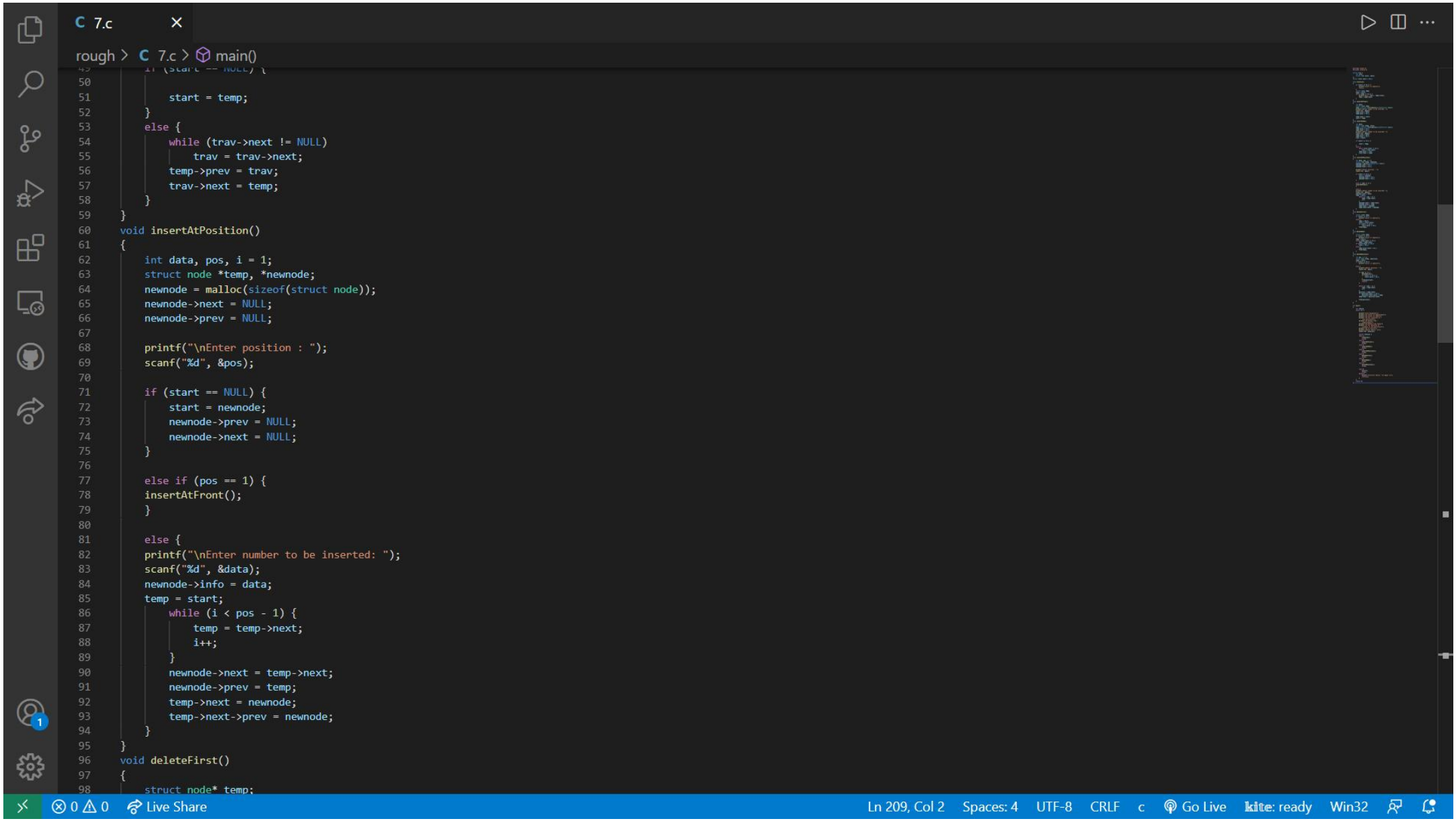
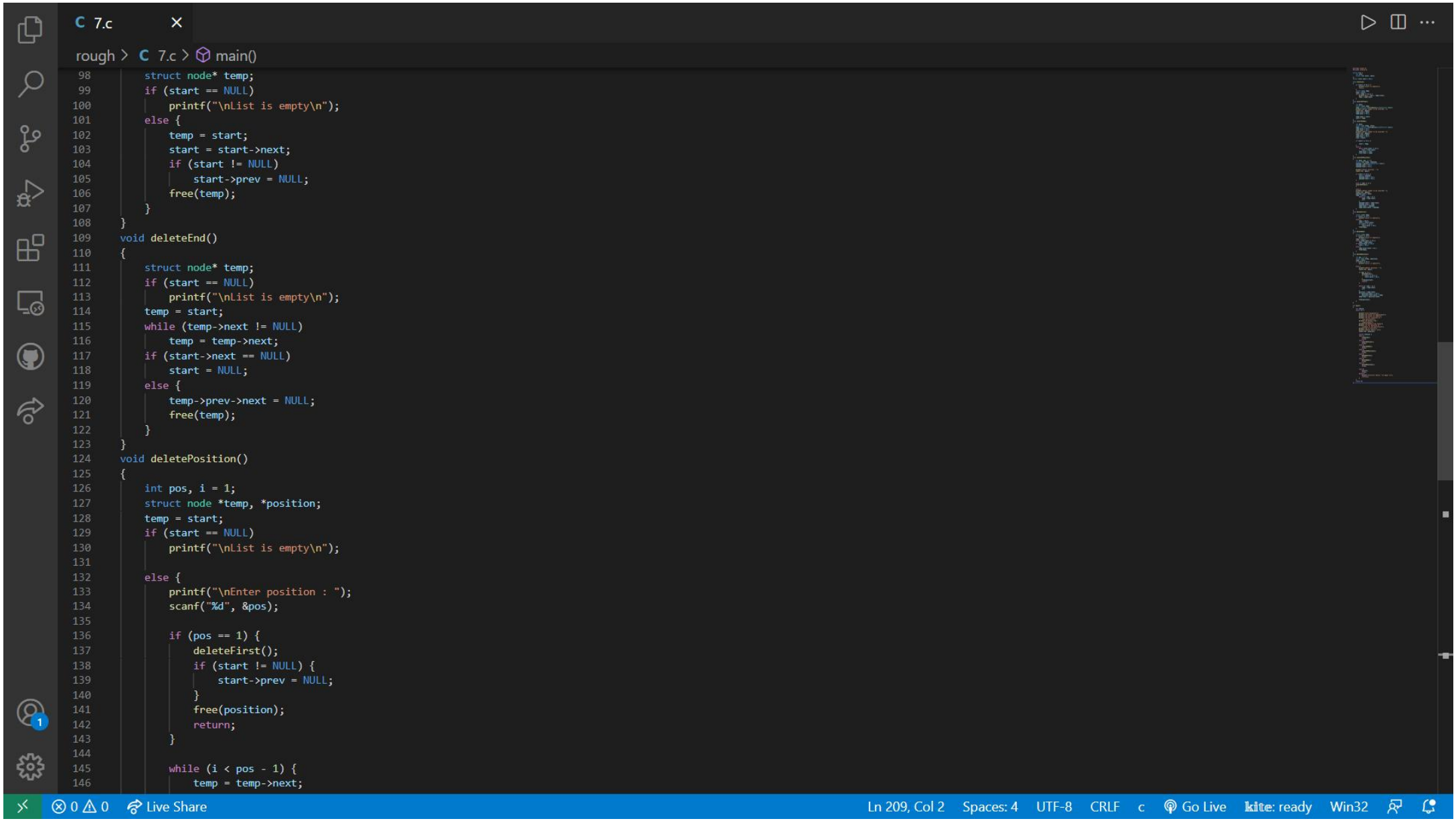
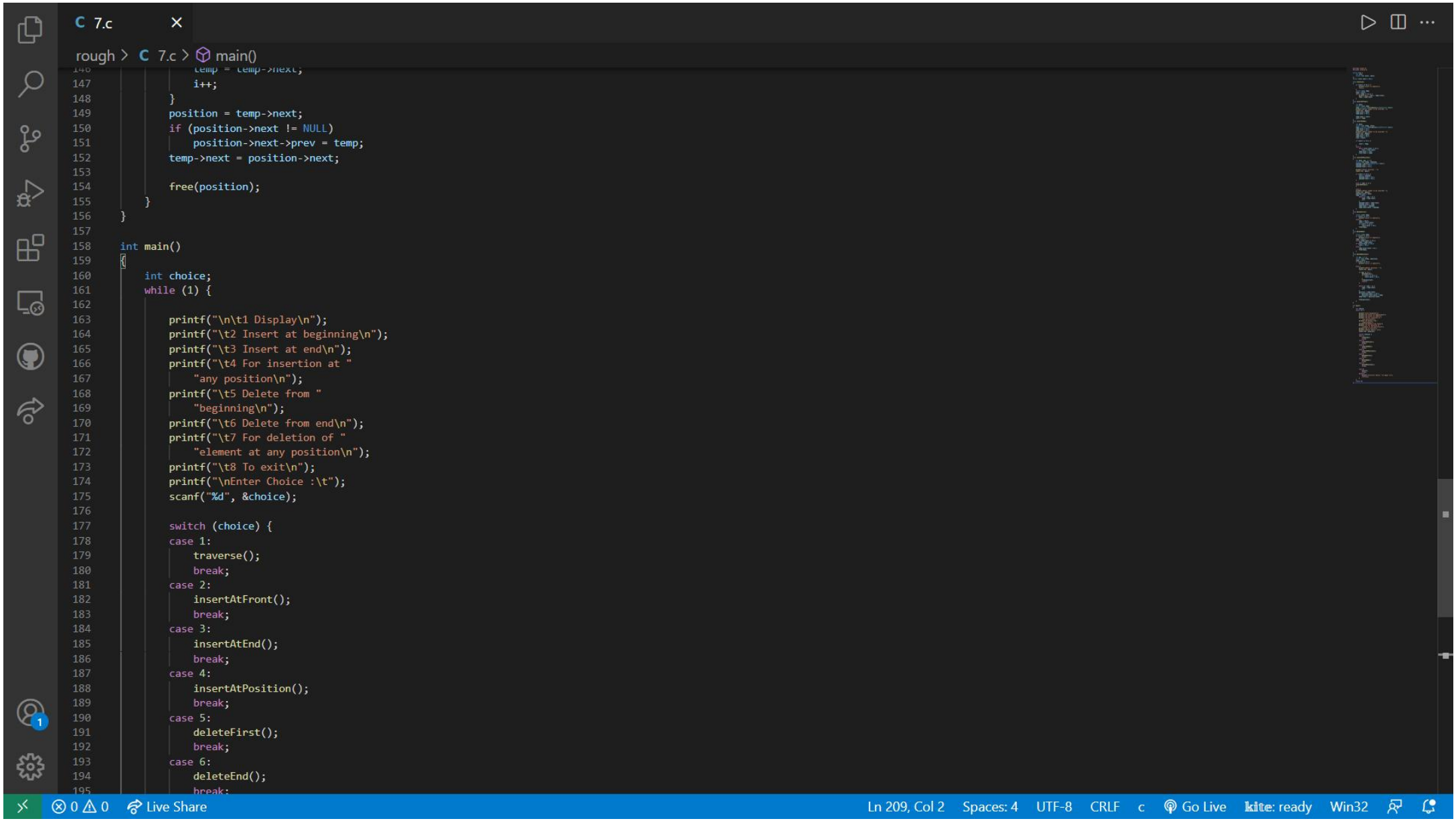
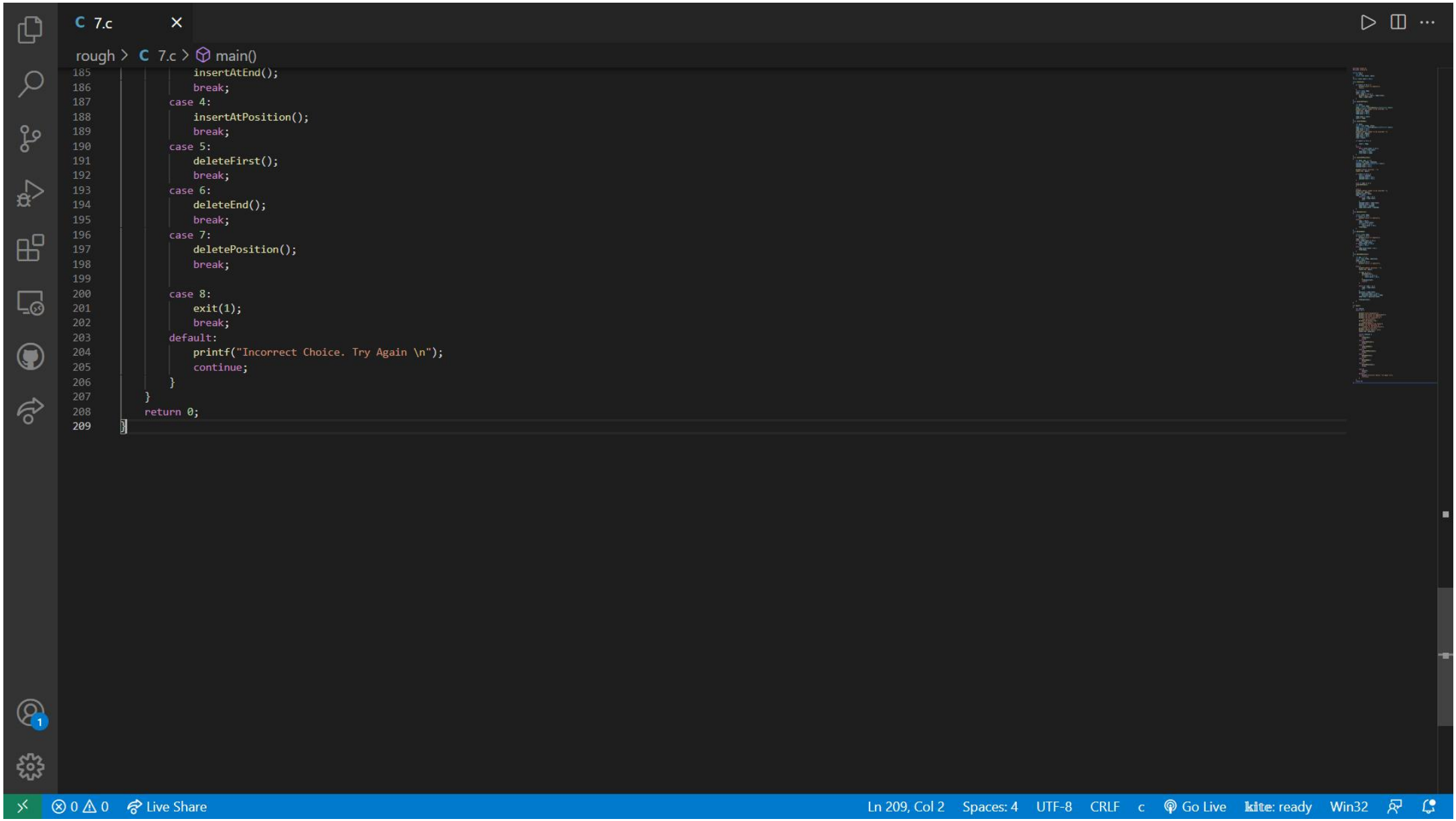
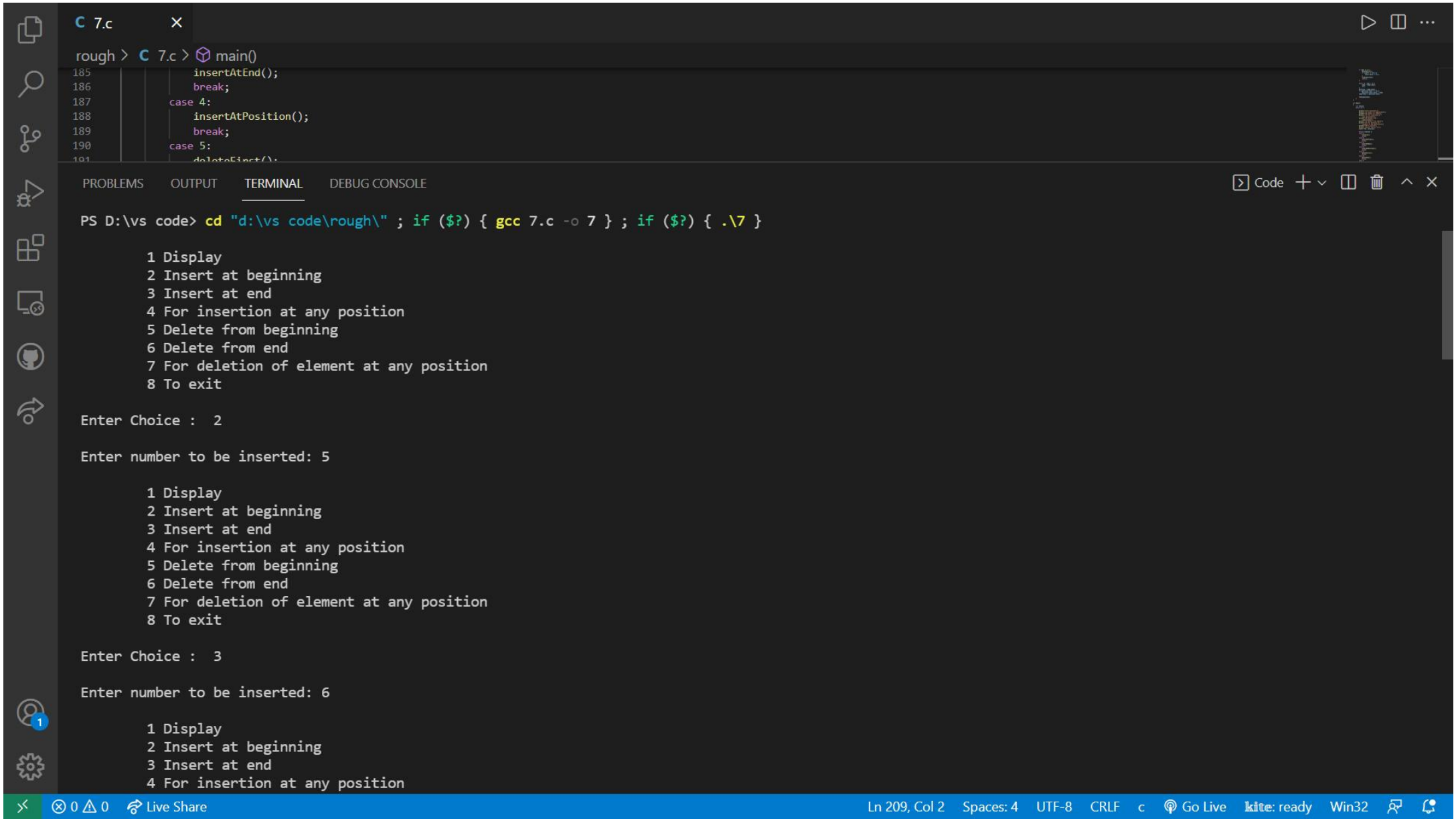

7.c
rough > 7.c > main()
#include <stdio.h>
#include <stdlib.h>
struct node {
int info;
struct node *prev, *next;
};
struct node* start = NULL;
void traverse()
{
if (start == NULL) {
printf("\nList is empty\n");
return;
}
struct node* temp;
temp = start;
while (temp != NULL) {
printf("Data = %d\n", temp->info);
temp = temp->next;
}
}
void insertAtFront()
{
int data;
struct node* temp;
temp = (struct node*)malloc(sizeof(struct node));
printf("\nEnter number to be inserted: ");
scanf("%d", &data);
temp->info = data;
temp->prev = NULL;
temp->next = start;
start = temp;
}
void insertAtEnd()
{
int data;
struct node *temp, *trav;
temp = (struct node*)malloc(sizeof(struct node));
temp->prev = NULL;
temp->next = NULL;
printf("\nEnter number to be inserted: ");
scanf("%d", &data);
temp->info = data;
temp->next = NULL;
trav = start;
if (start == NULL) {
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32

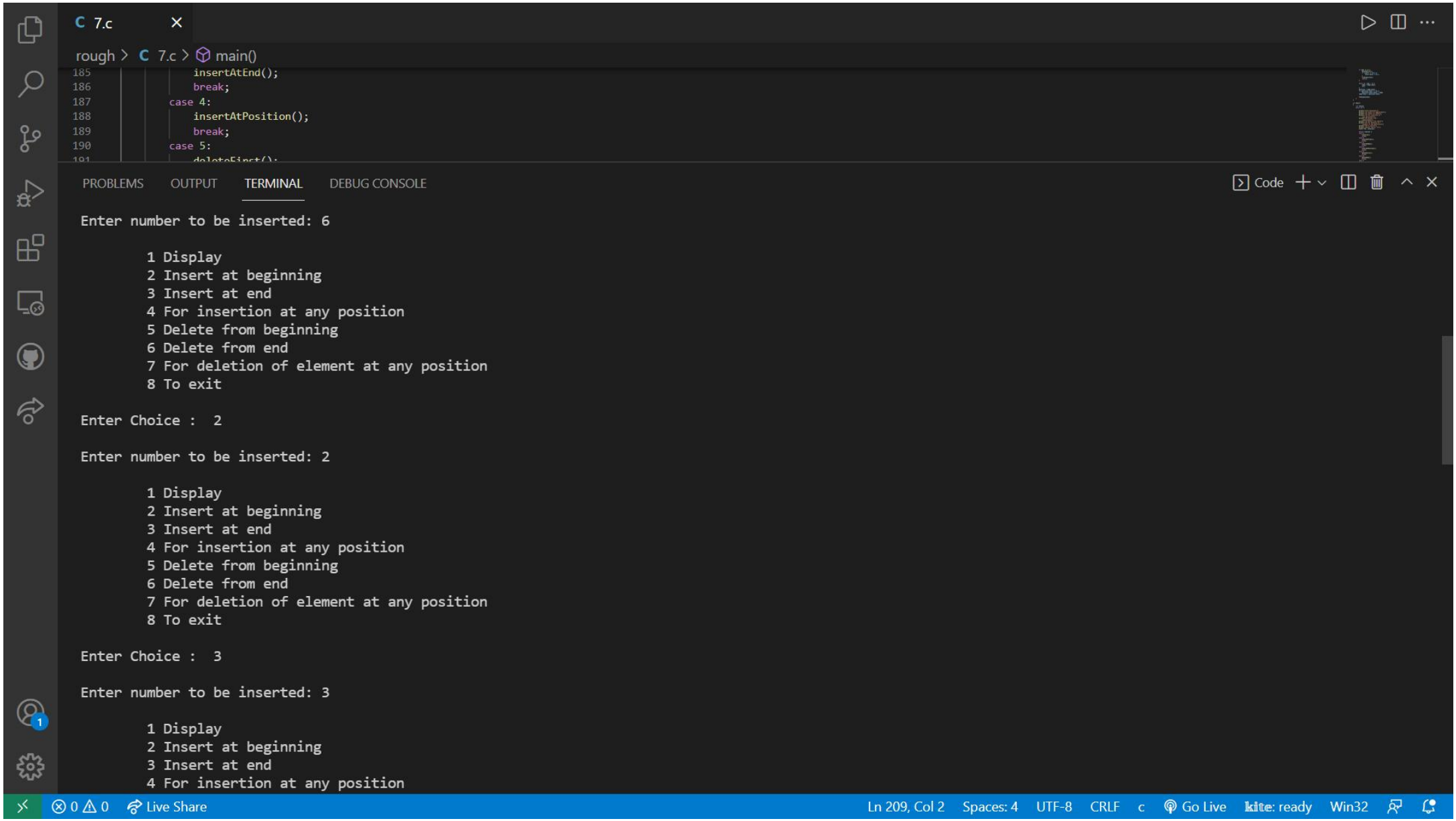

7.c
rough > 7.c > main()
start = temp;
}
else {
while (trav->next != NULL)
trav = trav->next;
temp->prev = trav;
trav->next = temp;
}
}
void insertAtPosition()
{
int data, pos, i = 1;
struct node *temp, *newnode;
newnode = malloc(sizeof(struct node));
newnode->next = NULL;
newnode->prev = NULL;
printf("\nEnter position : ");
scanf("%d", &pos);
if (start == NULL) {
start = newnode;
newnode->prev = NULL;
newnode->next = NULL;
}
else if (pos == 1) {
insertAtFront();
}
else {
printf("\nEnter number to be inserted: ");
scanf("%d", &data);
newnode->info = data;
temp = start;
while (i < pos - 1) {
temp = temp->next;
i++;
}
newnode->next = temp->next;
newnode->prev = temp;
temp->next = newnode;
temp->next->prev = newnode;
}
}
void deleteFirst()
{
struct node* temp;
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32

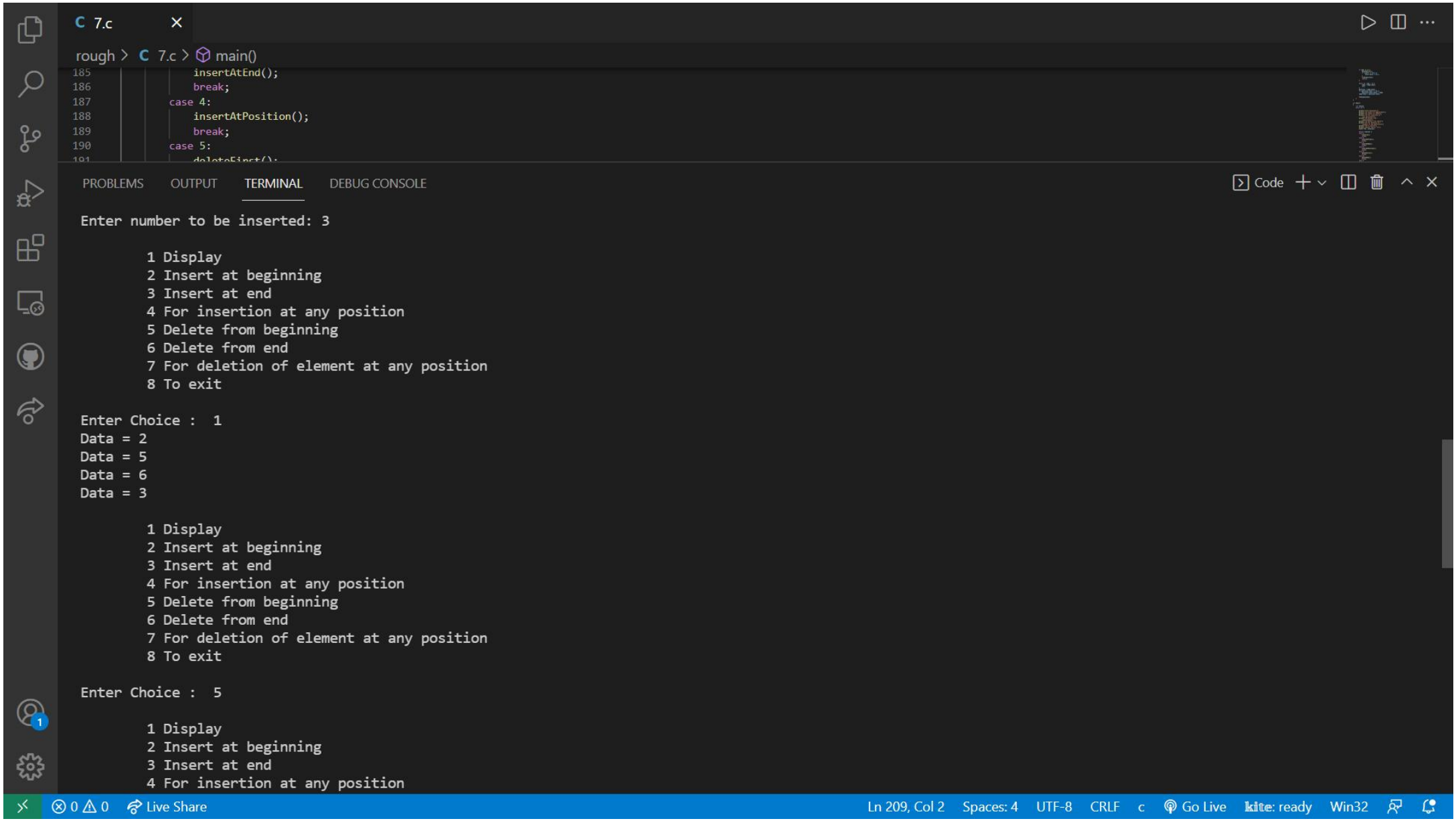

7.c
rough > 7.c > main()
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
else {
temp = start;
start = start->next;
if (start != NULL)
start->prev = NULL;
free(temp);
}
}
void deleteEnd()
{
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
temp = start;
while (temp->next != NULL)
temp = temp->next;
if (start->next == NULL)
start = NULL;
else {
temp->prev->next = NULL;
free(temp);
}
}
void deletePosition()
{
int pos, i = 1;
struct node *temp, *position;
temp = start;
if (start == NULL)
printf("\nList is empty\n");
else {
printf("\nEnter position : ");
scanf("%d", &pos);
if (pos == 1) {
deleteFirst();
if (start != NULL) {
start->prev = NULL;
}
free(position);
return;
}
while (i < pos - 1) {
temp = temp->next;
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32

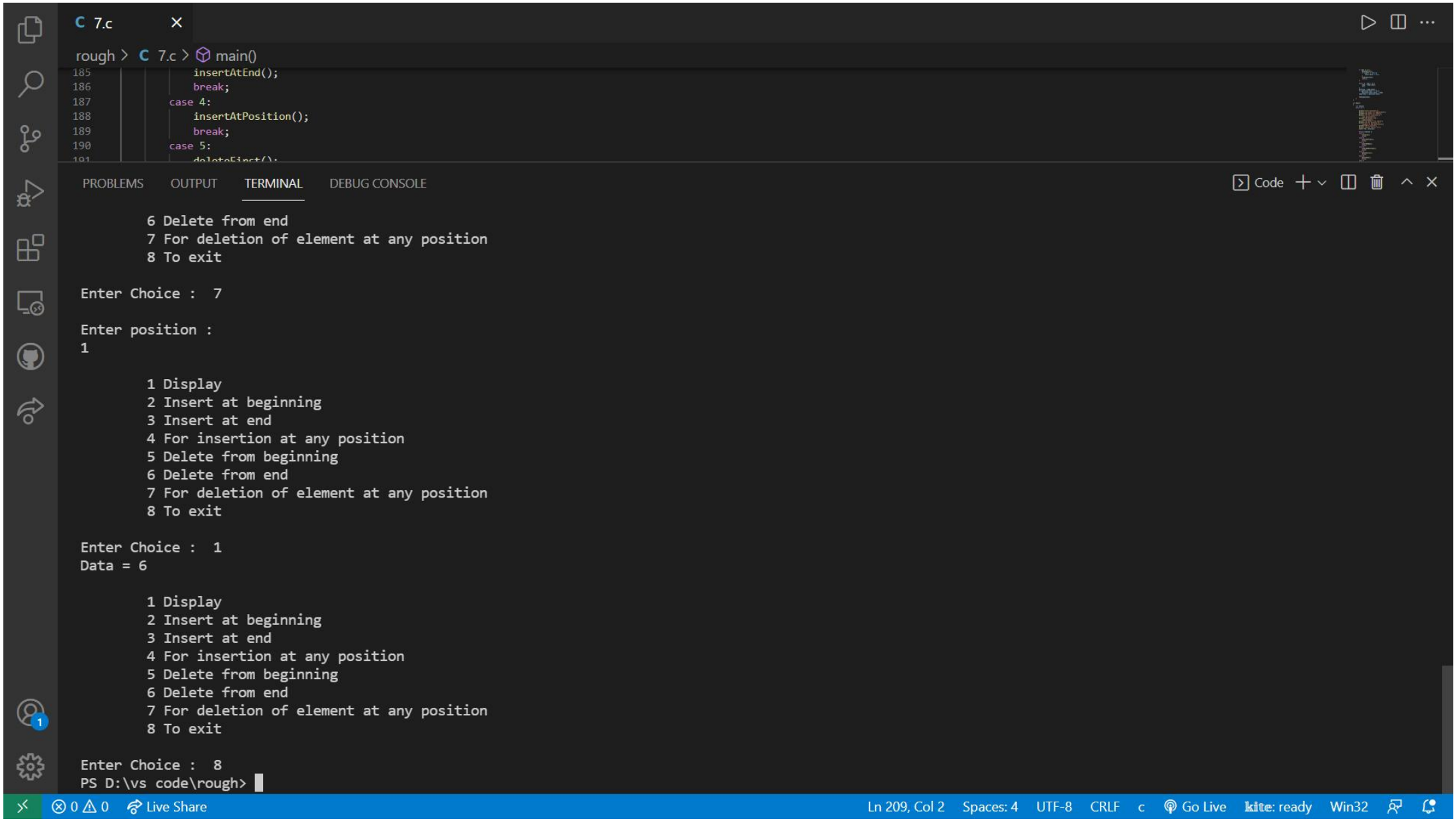

7.c
rough > C 7.c > main()
temp = temp->next;
i++;
}
position = temp->next;
if (position->next != NULL)
position->next->prev = temp;
temp->next = position->next;
free(position);
}
}
int main()
{
int choice;
while (1) {
printf("\n\t1 Display\n");
printf("\t2 Insert at beginning\n");
printf("\t3 Insert at end\n");
printf("\t4 For insertion at "
"any position\n");
printf("\t5 Delete from "
"beginning\n");
printf("\t6 Delete from end\n");
printf("\t7 For deletion of "
"element at any position\n");
printf("\t8 To exit\n");
printf("\nEnter Choice :\t");
scanf("%d", &choice);
switch (choice) {
case 1:
traverse();
break;
case 2:
insertAtFront();
break;
case 3:
insertAtEnd();
break;
case 4:
insertAtPosition();
break;
case 5:
deleteFirst();
break;
case 6:
deleteEnd();
break;
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32

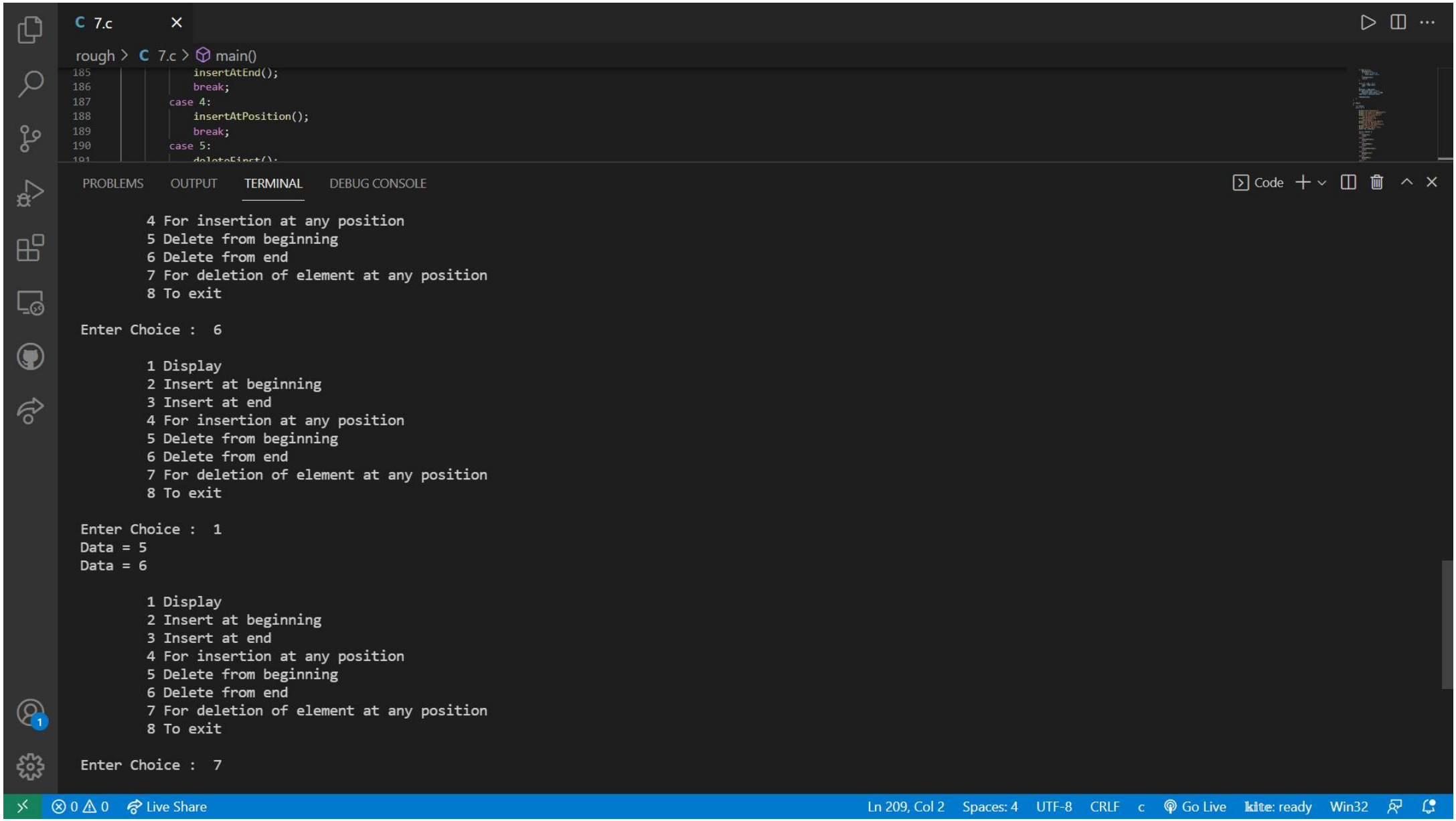

7.c
rough > C 7.c > main()
insertAtEnd();
break;
case 4:
insertAtPosition();
break;
case 5:
deleteFirst();
break;
case 6:
deleteEnd();
break;
case 7:
deletePosition();
break;
case 8:
exit(1);
break;
default:
printf("Incorrect Choice. Try Again \n");
continue;
}
}
return 0;
}
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32


7.c
rough > C 7.c > main()
insertAtEnd();
break;
case 4:
insertAtPosition();
break;
case 5:
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
PS D:\vs code> cd "d:\vs code\rough\" ; if ($?) { gcc 7.c -o 7 } ; if ($?) { .\7 }
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 2
Enter number to be inserted: 5
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 3
Enter number to be inserted: 6
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32


7.c
rough > C 7.c > main()
185 insertAtEnd();
186 break;
187 case 4:
188 insertAtPosition();
189 break;
190 case 5:
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
Enter number to be inserted: 6
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 2
Enter number to be inserted: 2
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 3
Enter number to be inserted: 3
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32


7.c
rough > C 7.c > main()
185 insertAtEnd();
186 break;
187 case 4:
188 insertAtPosition();
189 break;
190 case 5:
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
Enter number to be inserted: 3
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 1
Data = 2
Data = 5
Data = 6
Data = 3
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 5
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32


7.c
rough > C 7.c > main()
insertAtEnd();
break;
case 4:
insertAtPosition();
break;
case 5:
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 7
Enter position :
1
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 1
Data = 6
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 8
PS D:\vs code\rough>
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32


7.c
rough > C 7.c > main()
185 insertAtEnd();
186 break;
187 case 4:
188 insertAtPosition();
189 break;
190 case 5:
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 6
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 1
Data = 5
Data = 6
1 Display
2 Insert at beginning
3 Insert at end
4 For insertion at any position
5 Delete from beginning
6 Delete from end
7 For deletion of element at any position
8 To exit
Enter Choice : 7
0 0
Live Share
Ln 209, Col 2
Spaces: 4
UTF-8
CRLF
c
Go Live
kite: ready
Win32