Experiment No 03

Aim: Implementation of singly linked list / circular singly linked list and various operations for real-world.

Objectives:

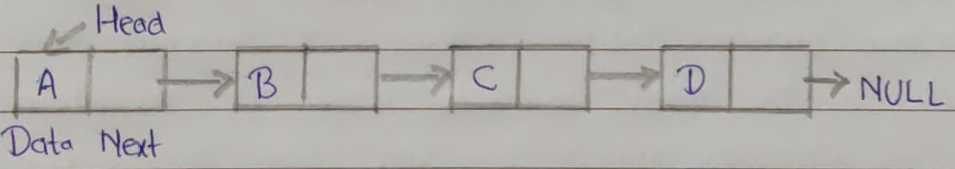
① To learn the basic principles of programming as applied to complex data structures.

② To learn the principles of linked list and its various operations.

Theory:

Introduction to linked list -

- A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.
- The elements in a linked list are linked using pointers.

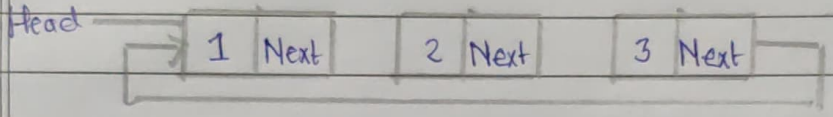


- In simple words, a linked list consists of nodes where each node contains a data field and a reference (link) to the next node in the list.

- **Singly linked list:** It is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type. The node contains a pointer to the next node means that the node stores the address of the next node in the sequence.

Introduction to linked list (Circular):

- In a circular linked list, the last node of the list contains a pointer to the first node of the list.
- We can have circular singly linked list as well as circular doubly linked list.
- We traverse a circular singly linked list until we reach the same node where we started.
- The circular linked list has no beginning and no ending.
- There is no null value present in the next part of any of the nodes.



- Circular linked lists are mostly used in task maintenance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfing where a record of pages visited in the past by user, is maintained in the form of circular linked list and can be accessed on clicking the previous button.

Insertion: The insertion into a singly linked list can be performed at different positions. Based on the positions of the new node being inserted, the insertion is categorized into the following categories.

1. Insertion at beginning - It involves inserting any element at the front of the list.

2. Insertion at the end of the list - It involves insertion at the last of the ~~end~~ linked list. The new node can be inserted as the only node in the list or it can be inserted as the last node.

3. Insertion after specified node - It involves insertion after the specified node of the linked list. We need to skip the desired number of nodes in order to reach after ~~the~~ which the new node will be inserted.

Deletion: The deletion of a node from a singly linked list can be performed at different position. Based on the position of node being deleted, the operation is categorized as -

1. Deletion at ~~beginning~~ end - It involves deleting the last node of the list.
2. Deletion at beginning - It involves deletion of node from the beginning of linked list.
3. Deletion after specific node - It involves deleting the node after the specified node in the list.

Traversing: In traversing, we simply visit each node of the list at least one in order to perform some specific operation in it.

Algorithm -

Insertion in the beginning -

Step 1 - If ptr = NULL
Write overflow
Goto step 7 [End of IF]

Step 2 - Set New_Node = Ptr
Step 3 - Set ptr = ptr → next
Step 4 - Set new_node → data = val
Step 5 - Set new_node → next = Head
Step 6 - Set head = new_node
Step 7 - Exit.

Insertion at the end
Step 1 - If ptr = NULL Write Overflow
Go to step 1
[End of IF]
Step 2 - Set new_node = ptr
Step 3 - Set ptr = ptr → next
Step 4 - Set new_node → data = val
Step 5 - Set new_node → next = NULL
Step 6 - Set ptr = head
Step 7 - Repeat step 8 while ptr → NEXT != NULL
Step 8 - Set PTR = PTR → NEXT [End of loop]
Step 9 - Set ptr → next = new_node
Step 10 - Exit

Insertion at the specified node-
Step 1 - if ptr = null Write overflow
goto step 12 [End of IF]
Step 2 - Set new_node → ~~Data = val~~ ptr
Step 3 - Set new_node → Data = val
Step 4 - Set temp = head
Step 5 - Set i = 0
Step 6 - Repeat step 5 and 6 until i

Step 7 - Temp = Temp → next
Step 8 - If Temp = Null
Write "Desired node not present"
Goto step 12 [End of IF]
[End of loop]
Step 9 - PTR → NEXT = TEMP → NEXT
Step 10 - TEMP → NEXT = PTR
Step 11 - Set ptr = new_node
Step 12 - Exit

Deletion at beginning -
Step 1 - If head = null
Write underflow
Goto step 5 [End of IF]
Step 2 - Set ptr = head
Step 3 - Set head = head → next
Step 4 - Free ptr
Step 5 - Exit

Deletion at specified node -
Step 1 - If head = null
Write underflow
Goto step 10
[End of IF]
Step 2 - Set Temp = Head
Step 3 - Set i = 0
Step 4 - Repeat Step 5 to 8 until i
Step 5 - temp1 = temp
Step 6 - temp = temp → next

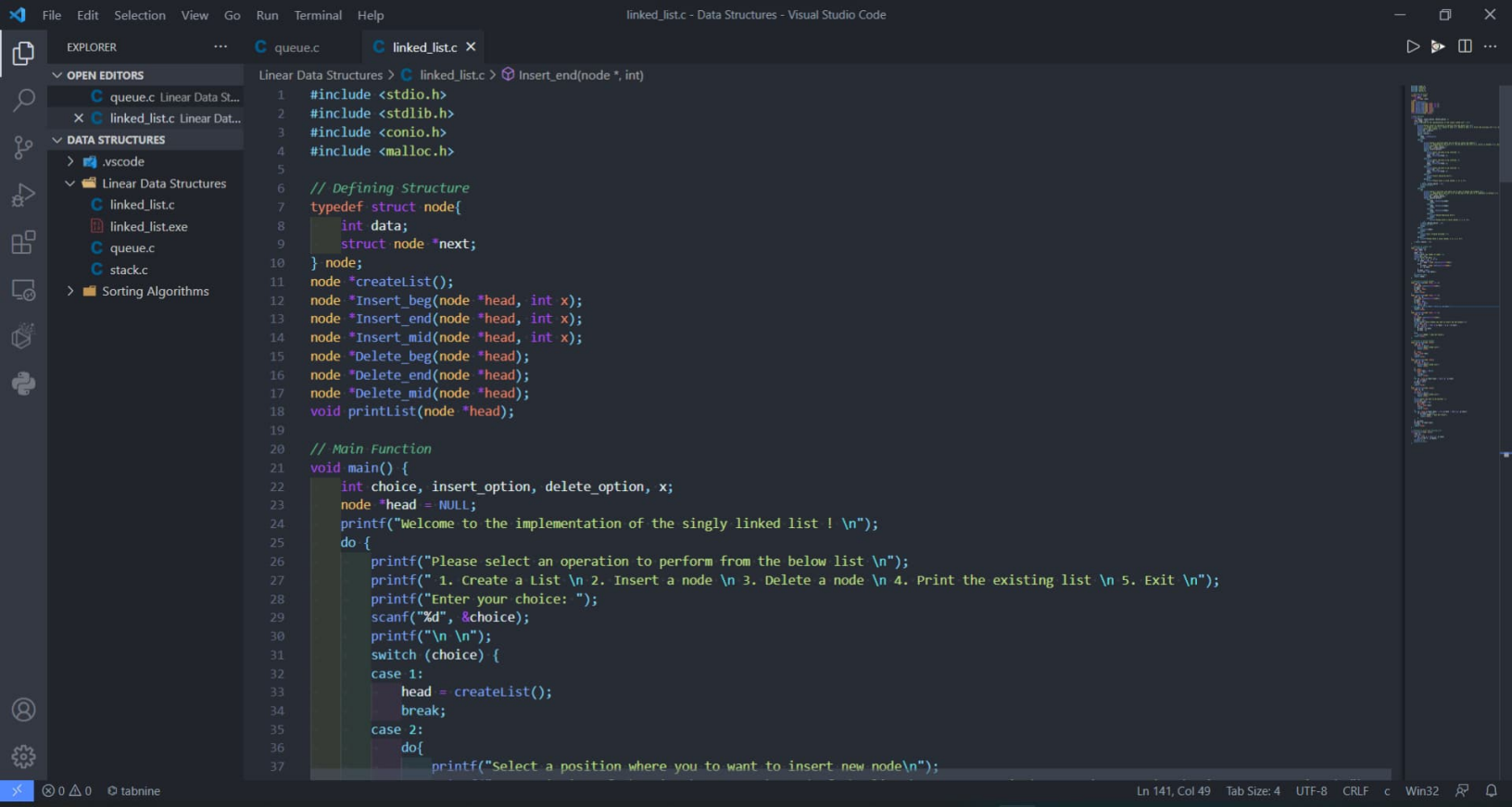
Step 7- If temp = null
Write "Desired node not present"
Goto step 12 [End of if]
Step 8- i = i+1
End of loop
Step 9- temp1 → next = temp → next
Step 10- FREE TEMP
Step 11- Exit

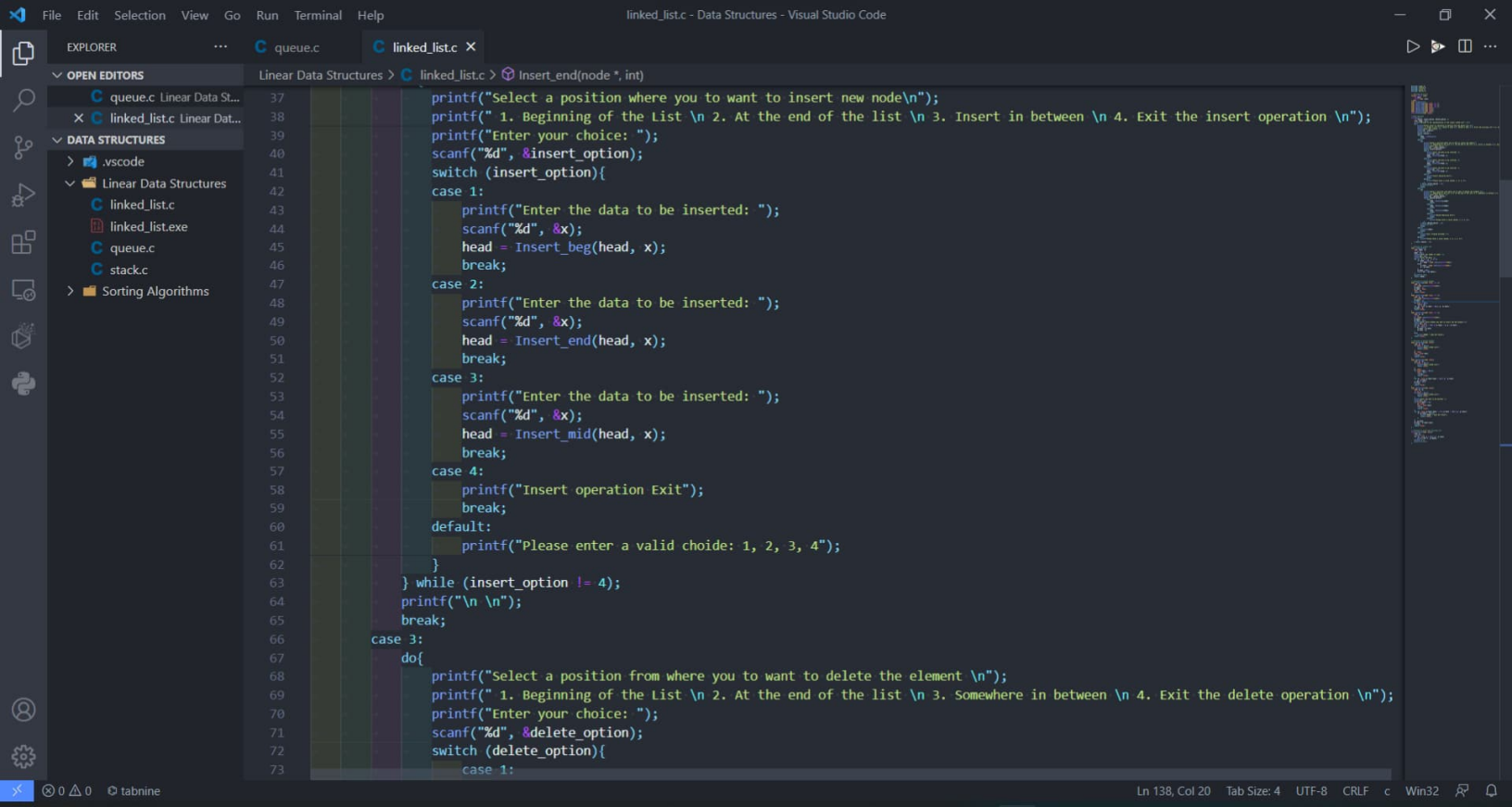
Deletion at the end:
Step 1- If head = null
Write underflow
goto step 8 [end of if]
Step 2- set ptr = head
Step 3- Repeat step 4 and 5 while ptr → next[] = null
Step 4- Set PREPTR = PTR
Step 5- Set PTR = PTR → NEXT
[End of loop]
Step 6- Set PREPTR → NEXT = NULL
Step 7- FREE PTR
Step 8- Exit.

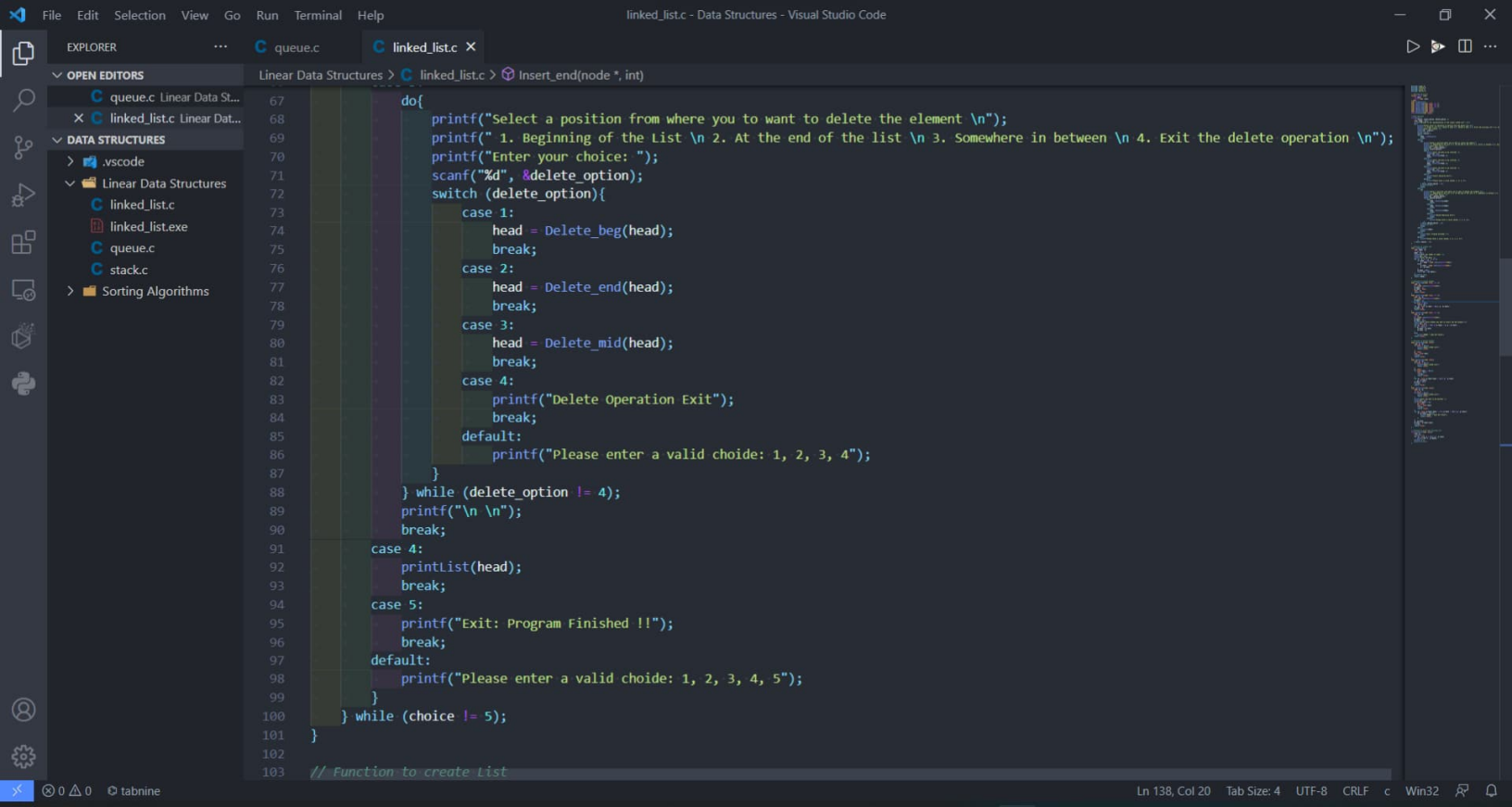
Examples-
(1) List of images that need to be burned to a CD in a medical imaging application.
(2) List of objects in a 3D game that need to be rendered to the screen.
(3) Songs in music player are linked to previous and next song, you can play songs either from starting or ending.

Conclusion: In this experiment, we learnt about linked list. (singly and circular). We implemented linked list and its operations. ~~&~~ like insertion and deletion. We also came across the examples where ~~st~~ linked lists are implemented in real-world.

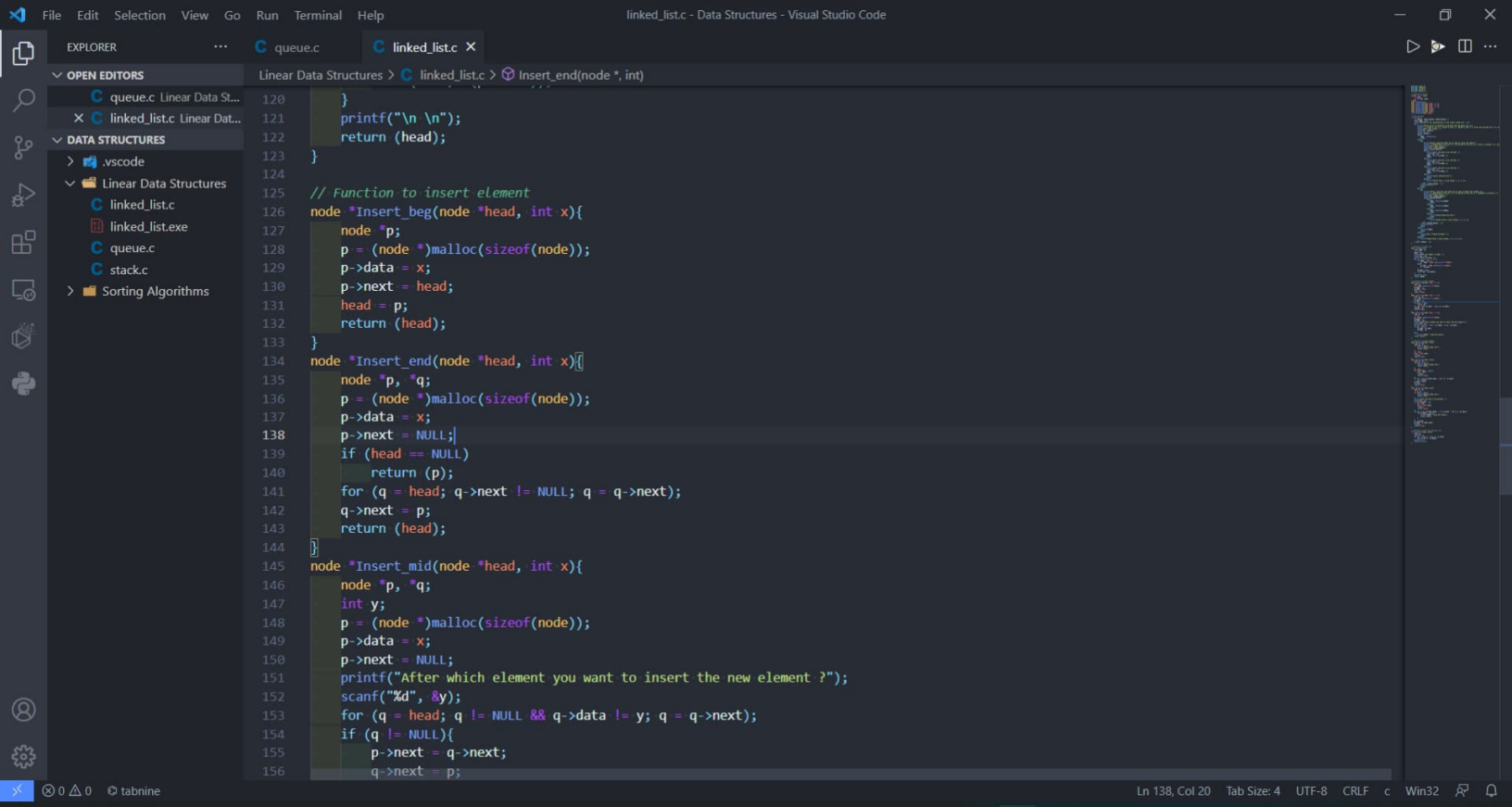
Outcome: Applied the concepts of singly, circular and doubly linked list for real-world applications.

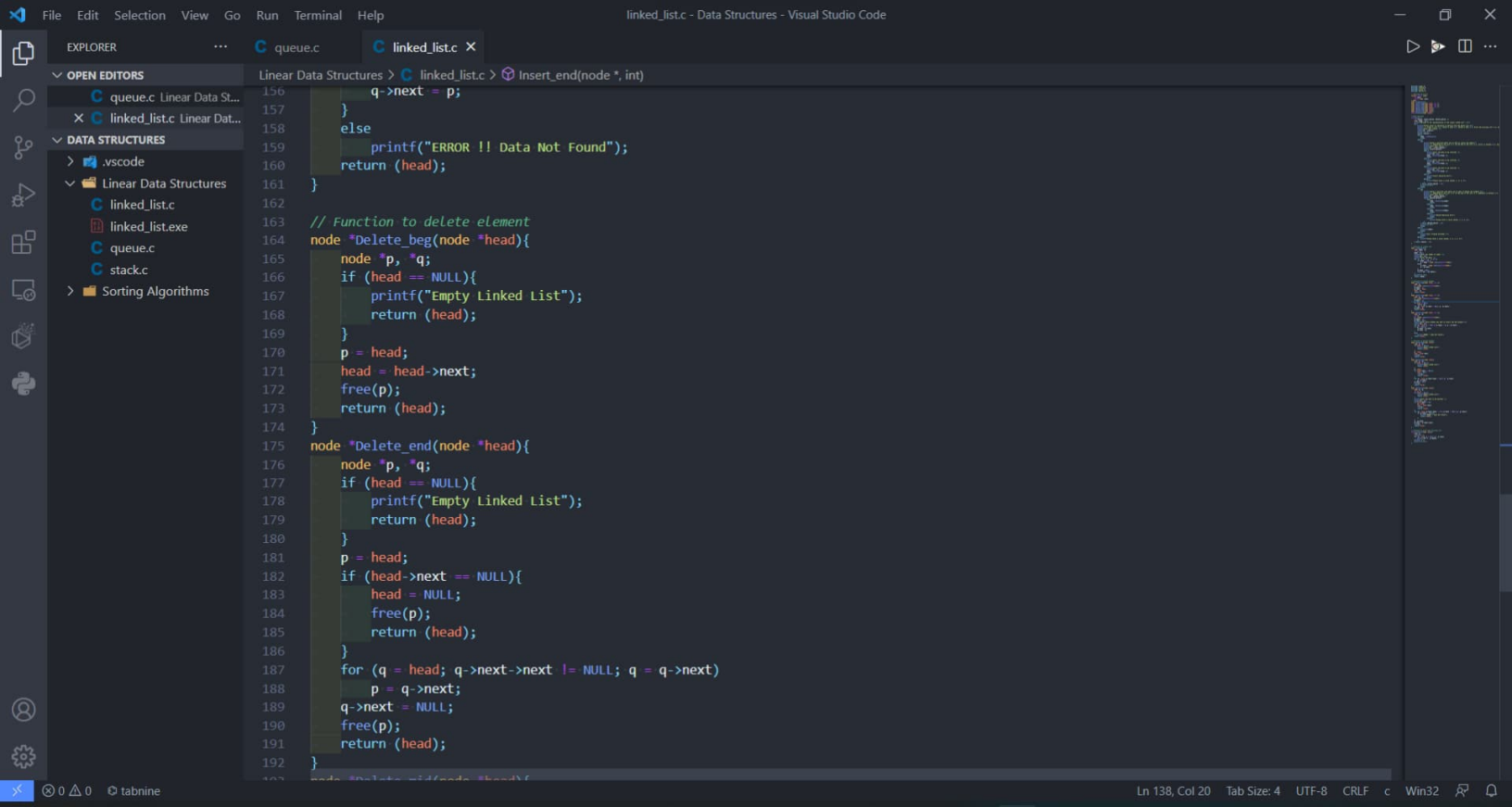

File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
// Defining Structure
typedef struct node{
int data;
struct node *next;
} node;
node *createList();
node *Insert_beg(node *head, int x);
node *Insert_end(node *head, int x);
node *Insert_mid(node *head, int x);
node *Delete_beg(node *head);
node *Delete_end(node *head);
node *Delete_mid(node *head);
void printList(node *head);
// Main Function
void main() {
int choice, insert_option, delete_option, x;
node *head = NULL;
printf("Welcome to the implementation of the singly linked list ! \n");
do {
printf("Please select an operation to perform from the below list \n");
printf(" 1. Create a List \n 2. Insert a node \n 3. Delete a node \n 4. Print the existing list \n 5. Exit \n");
printf("Enter your choice: ");
scanf("%d", &choice);
printf("\n \n");
switch (choice) {
case 1:
head = createList();
break;
case 2:
do{
printf("Select a position where you to want to insert new node\n");
tabnine
Ln 141, Col 49
Tab Size: 4
UTF-8
CRLF
c
Win32

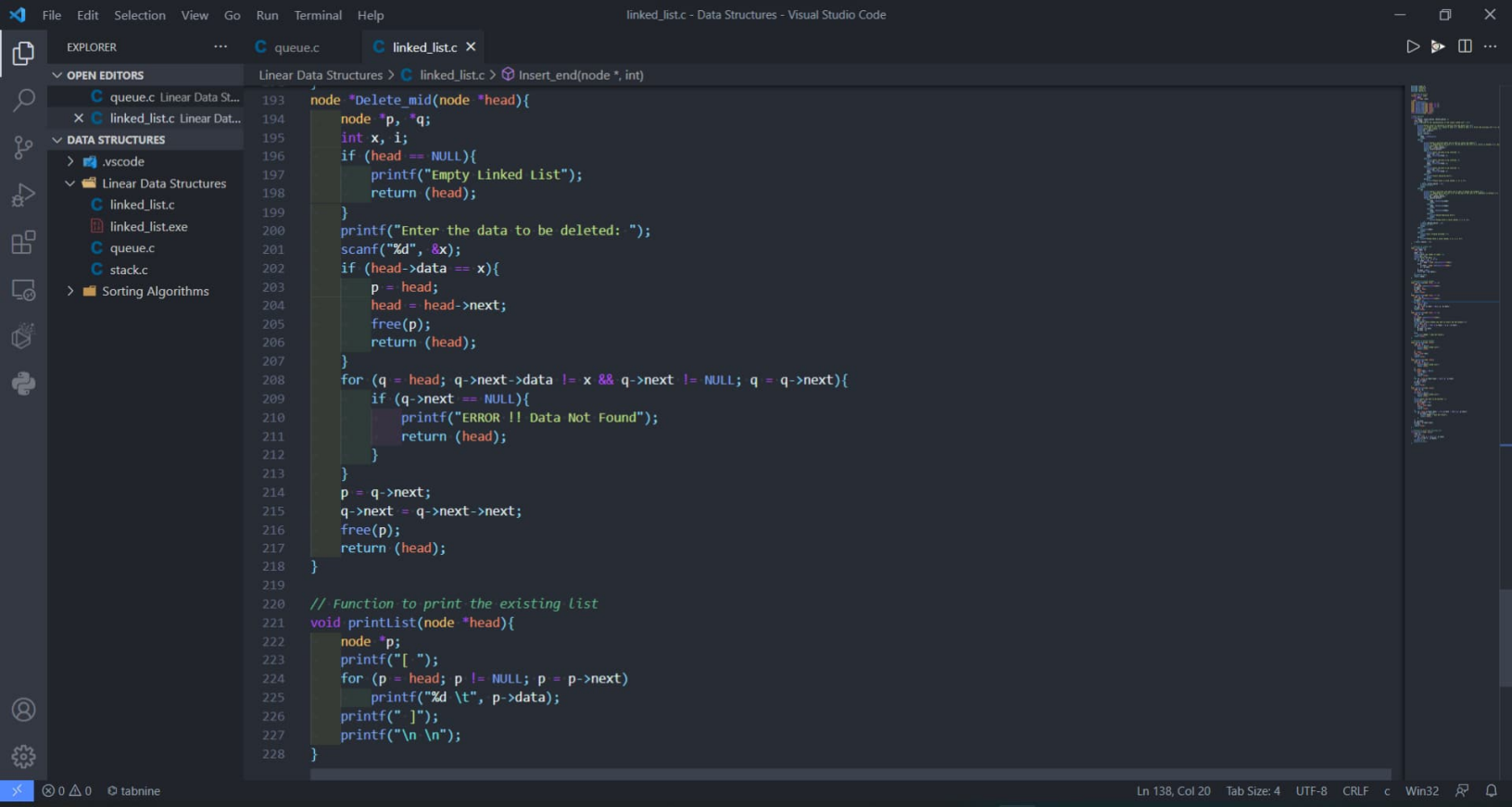

File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
printf("Select a position where you to want to insert new node\n");
printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Insert in between \n 4. Exit the insert operation \n");
printf("Enter your choice: ");
scanf("%d", &insert_option);
switch (insert_option){
case 1:
printf("Enter the data to be inserted: ");
scanf("%d", &x);
head = Insert_beg(head, x);
break;
case 2:
printf("Enter the data to be inserted: ");
scanf("%d", &x);
head = Insert_end(head, x);
break;
case 3:
printf("Enter the data to be inserted: ");
scanf("%d", &x);
head = Insert_mid(head, x);
break;
case 4:
printf("Insert operation Exit");
break;
default:
printf("Please enter a valid choide: 1, 2, 3, 4");
}
} while (insert_option != 4);
printf("\n \n");
break;
case 3:
do{
printf("Select a position from where you to want to delete the element \n");
printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Somewhere in between \n 4. Exit the delete operation \n");
printf("Enter your choice: ");
scanf("%d", &delete_option);
switch (delete_option){
case 1:
tabnine
Ln 138, Col 20
Tab Size: 4
UTF-8
CRLF
c
Win32

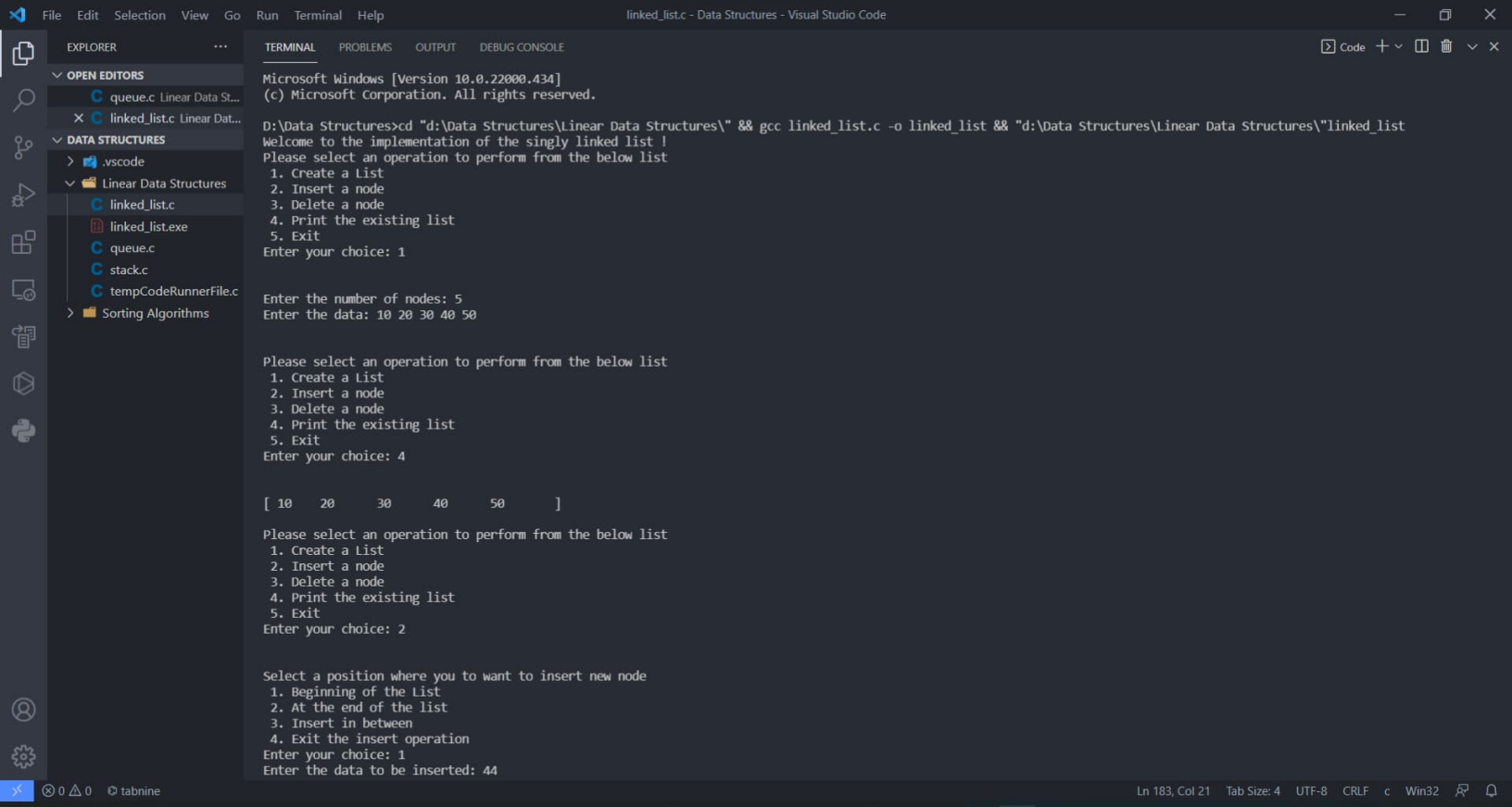

File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
do{
printf("Select a position from where you to want to delete the element \n");
printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Somewhere in between \n 4. Exit the delete operation \n");
printf("Enter your choice: ");
scanf("%d", &delete_option);
switch (delete_option){
case 1:
head = Delete_beg(head);
break;
case 2:
head = Delete_end(head);
break;
case 3:
head = Delete_mid(head);
break;
case 4:
printf("Delete Operation Exit");
break;
default:
printf("Please enter a valid choide: 1, 2, 3, 4");
}
} while (delete_option != 4);
printf("\n \n");
break;
case 4:
printList(head);
break;
case 5:
printf("Exit: Program Finished !!");
break;
default:
printf("Please enter a valid choide: 1, 2, 3, 4, 5");
}
} while (choice != 5);
}
// Function to create List
tabnine
Ln 138, Col 20
Tab Size: 4
UTF-8
CRLF
c
Win32

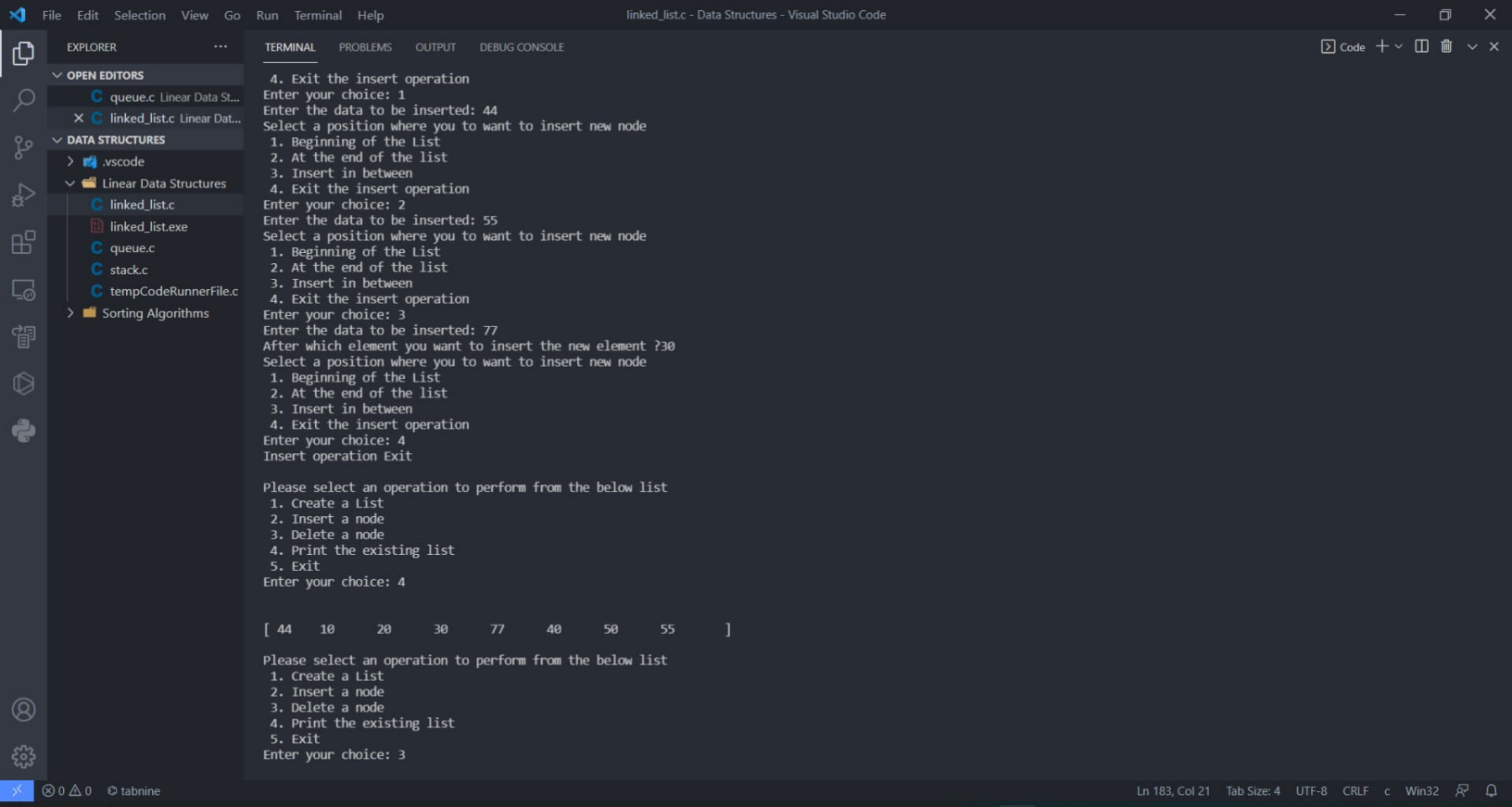
```
97          default:
98              printf("Please enter a valid choide: 1, 2, 3, 4, 5");
99          }
100     } while (choice != 5);
101 }
102
103 // Function to create List
104 node *createList(){
105     node *head, *p;
106     int i, n;
107     head = NULL;
108     printf("Enter the number of nodes: ");
109     scanf("%d", &n);
110     printf("Enter the data: ");
111     for (i = 0; i <= n - 1; i++){
112         if (head == NULL)
113             p = head = (node *)malloc(sizeof(node));
114         else{
115             p->next = (node *)malloc(sizeof(node));
116             p = p->next;
117         }
118         p->next = NULL;
119         scanf("%d", &(p->data));
120     }
121     printf("\n \n");
122     return (head);
123 }
124
125 // Function to insert element
126 node *Insert_beg(node *head, int x){
127     node *p;
128     p = (node *)malloc(sizeof(node));
129     p->data = x;
130     p->next = head;
131     head = p;
132     return (head);
133 }
```

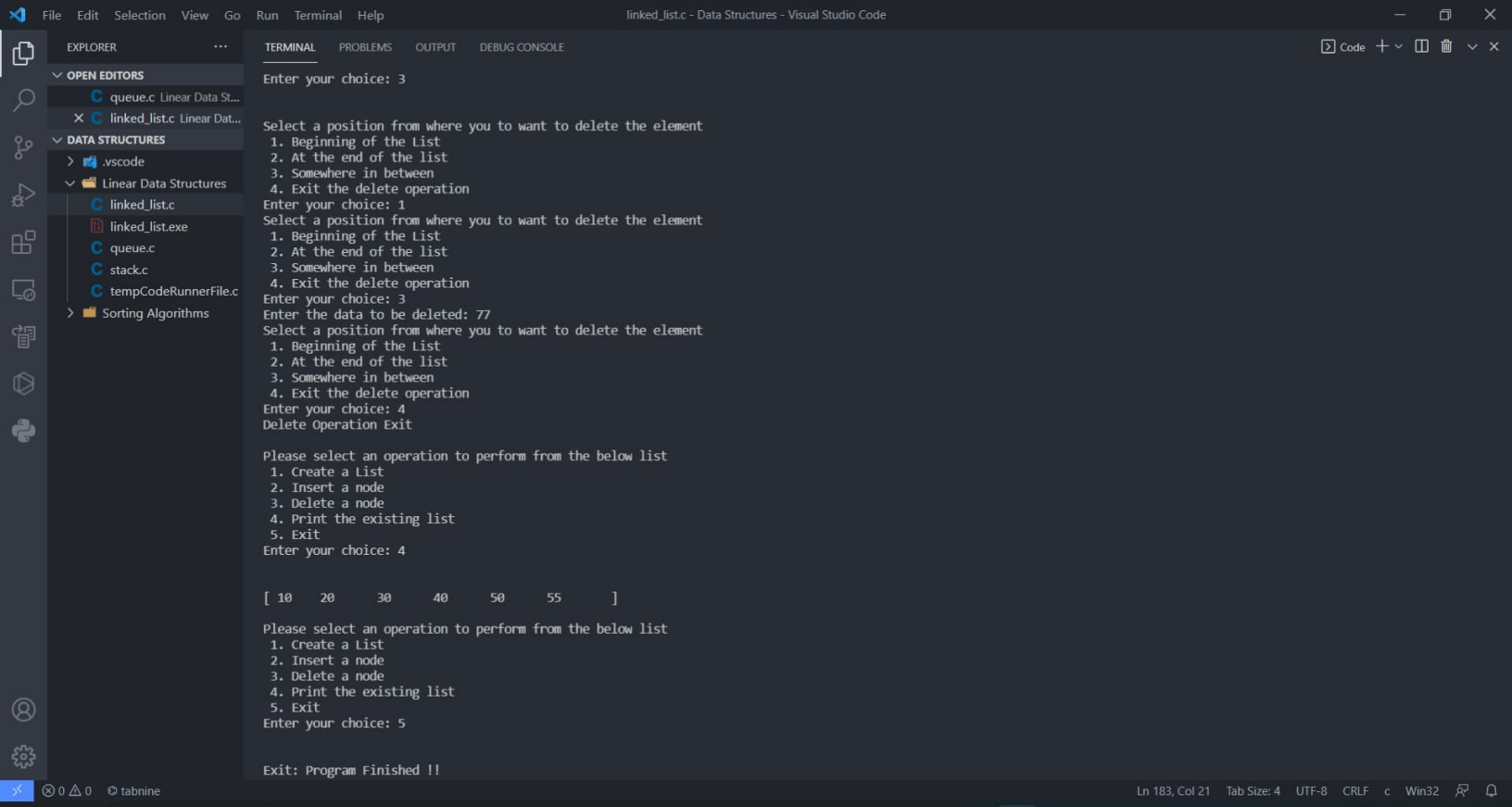

File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
120 }
121 printf("\n \n");
122 return (head);
123 }
124
125 // Function to insert element
126 node *Insert_beg(node *head, int x){
127 node *p;
128 p = (node *)malloc(sizeof(node));
129 p->data = x;
130 p->next = head;
131 head = p;
132 return (head);
133 }
134 node *Insert_end(node *head, int x){
135 node *p, *q;
136 p = (node *)malloc(sizeof(node));
137 p->data = x;
138 p->next = NULL;
139 if (head == NULL)
140 return (p);
141 for (q = head; q->next != NULL; q = q->next);
142 q->next = p;
143 return (head);
144 }
145 node *Insert_mid(node *head, int x){
146 node *p, *q;
147 int y;
148 p = (node *)malloc(sizeof(node));
149 p->data = x;
150 p->next = NULL;
151 printf("After which element you want to insert the new element ?");
152 scanf("%d", &y);
153 for (q = head; q != NULL && q->data != y; q = q->next);
154 if (q != NULL){
155 p->next = q->next;
156 q->next = p;
tabnine
Ln 138, Col 20
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
q->next = p;
}
else
printf("ERROR !! Data Not Found");
return (head);
}
// Function to delete element
node *Delete_beg(node *head){
node *p, *q;
if (head == NULL){
printf("Empty Linked List");
return (head);
}
p = head;
head = head->next;
free(p);
return (head);
}
node *Delete_end(node *head){
node *p, *q;
if (head == NULL){
printf("Empty Linked List");
return (head);
}
p = head;
if (head->next == NULL){
head = NULL;
free(p);
return (head);
}
for (q = head; q->next->next != NULL; q = q->next)
p = q->next;
q->next = NULL;
free(p);
return (head);
}
0 0 tabnine
Ln 138, Col 20 Tab Size: 4 UTF-8 CRLF c Win32


File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
queue.c
linked_list.c
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
Sorting Algorithms
Linear Data Structures > linked_list.c > Insert_end(node *, int)
193 node *Delete_mid(node *head){
194 node *p, *q;
195 int x, i;
196 if (head == NULL){
197 printf("Empty Linked List");
198 return (head);
199 }
200 printf("Enter the data to be deleted: ");
201 scanf("%d", &x);
202 if (head->data == x){
203 p = head;
204 head = head->next;
205 free(p);
206 return (head);
207 }
208 for (q = head; q->next->data != x && q->next != NULL; q = q->next){
209 if (q->next == NULL){
210 printf("ERROR !! Data Not Found");
211 return (head);
212 }
213 }
214 p = q->next;
215 q->next = q->next->next;
216 free(p);
217 return (head);
218 }
219
220 // Function to print the existing list
221 void printList(node *head){
222 node *p;
223 printf("[");
224 for (p = head; p != NULL; p = p->next)
225 printf("%d \t", p->data);
226 printf("]");
227 printf("\n \n");
228 }
tabnine
Ln 138, Col 20
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
tempCodeRunnerFile.c
Sorting Algorithms
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Microsoft Windows [Version 10.0.22000.434]
(c) Microsoft Corporation. All rights reserved.
D:\Data Structures>cd "d:\Data Structures\Linear Data Structures\" && gcc linked_list.c -o linked_list && "d:\Data Structures\Linear Data Structures\"linked_list
Welcome to the implementation of the singly linked list !
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 1
Enter the number of nodes: 5
Enter the data: 10 20 30 40 50
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 4
[10 20 30 40 50]
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 2
Select a position where you to want to insert new node
1. Beginning of the List
2. At the end of the list
3. Insert in between
4. Exit the insert operation
Enter your choice: 1
Enter the data to be inserted: 44
tabnine
Ln 183, Col 21 Tab Size: 4 UTF-8 CRLF c Win32


File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
tempCodeRunnerFile.c
Sorting Algorithms
TERMINAL
PROBLEMS
OUTPUT
DEBUG CONSOLE
Code
4. Exit the insert operation
Enter your choice: 1
Enter the data to be inserted: 44
Select a position where you to want to insert new node
1. Beginning of the List
2. At the end of the list
3. Insert in between
4. Exit the insert operation
Enter your choice: 2
Enter the data to be inserted: 55
Select a position where you to want to insert new node
1. Beginning of the List
2. At the end of the list
3. Insert in between
4. Exit the insert operation
Enter your choice: 3
Enter the data to be inserted: 77
After which element you want to insert the new element ?30
Select a position where you to want to insert new node
1. Beginning of the List
2. At the end of the list
3. Insert in between
4. Exit the insert operation
Enter your choice: 4
Insert operation Exit
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 4
[44 10 20 30 77 40 50 55]
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 3
0 0
tabnine
Ln 183, Col 21
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
linked_list.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
linked_list.c Linear Dat...
DATA STRUCTURES
.vscode
Linear Data Structures
linked_list.c
linked_list.exe
queue.c
stack.c
tempCodeRunnerFile.c
Sorting Algorithms
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Enter your choice: 3
Select a position from where you to want to delete the element
1. Beginning of the List
2. At the end of the list
3. Somewhere in between
4. Exit the delete operation
Enter your choice: 1
Select a position from where you to want to delete the element
1. Beginning of the List
2. At the end of the list
3. Somewhere in between
4. Exit the delete operation
Enter your choice: 3
Enter the data to be deleted: 77
Select a position from where you to want to delete the element
1. Beginning of the List
2. At the end of the list
3. Somewhere in between
4. Exit the delete operation
Enter your choice: 4
Delete Operation Exit
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 4
[10 20 30 40 50 55]
Please select an operation to perform from the below list
1. Create a List
2. Insert a node
3. Delete a node
4. Print the existing list
5. Exit
Enter your choice: 5
Exit: Program Finished !!
tabnine
Ln 183, Col 21 Tab Size: 4 UTF-8 CRLF c Win32