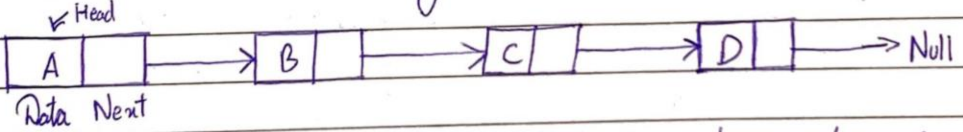
Experiment No. 5

* Aim :- Implementation of Singly Linked List / Circular Singly Linked List and various application for real world.

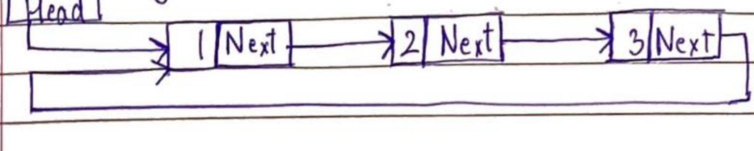
* Objective
1. To learn the basic principles of programming as applied to complex data structure.
2. To learn the principles of Linked list and its various operation.

* Theory
* Introduction to Linked List
⇒ A Linked list to a linear data structure, in which the elements are not stored at contigous memory locations. The element in a linked list are linked using pointers.



In simple words, a linked list consist of nodes where each node contains a data field and a reference to the next node.

* Introduction to ~~Cir~~ Circular Singly Linked List
-> In a circular singly 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 singly linked list has no beginning and no ending. There is no null value present in the next part of any nodes.



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

* Insertion

⇒ The insertion into a singly linked list can be performed at different position. Based on the position 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 linked list. The new node can be inserted as the only 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 linked list. We need to skip the desired number of nodes in order to reach the node after 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 : It involves deletion of a node from beginning
2. Deletion at end:- It involves deleting the last node of the list.
3. Deletion after specified node:- It involves deleting the node after the specified node in the list.

Traversing

→ In traversing, we simply visit eachnode of the list at least once in order to perform some specific operation on it.

Algorithm

Insertion in the begining

```
Step 1:- IF PTR=NULL
             Write OVERFLOW
              Go to 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 END

```
Step 1:- IF PTR=NULL Write OVERFLOW
             Goto Step1
             [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 specified node:-

```
Step 1:- IF PTR = NULL
            WRITE OVERFLOW
             Go To Step 12
            END OF IF
Step 2:- SET NEW_NODE → PTR
Step 3:- NEW_NODE → DATA = VAL
Step 4:- SET TEMP = HEAD
Step 5:- SET I=0
Step 6:- REPEAT ~~UNTIL~~ UNTIL I
Step 7:- TEMP = TEMP → NEXT
Step 8:- IF TEMP = NULL
           Write "DESIRED NODE NOT PRESENT"
                Go To 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 of 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 STEPS 5 TO 8 UNTIL 1
Step 5:-  TEMP 1 = 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:-  TEMP 1 => TEMP ->NEXT
Step 10:-  FREE TEMP
Step 11:-  EXIT.
```

* Deletion at the END

```
Step 1:- IF HEAD = NULL
          Write UNDERFLOW
          Go To STEP 8   [END OF IF]
Step 2:- SET PTR = HEAD
Step 3:- Repeat steps 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:-  FREEPTR
Step 8:-  EXIT.
```

★ Examples

→ 1) List of images that need to be burned to a CD in a medical imaging application.

2) List of users of a website that need to be emailed some notification.

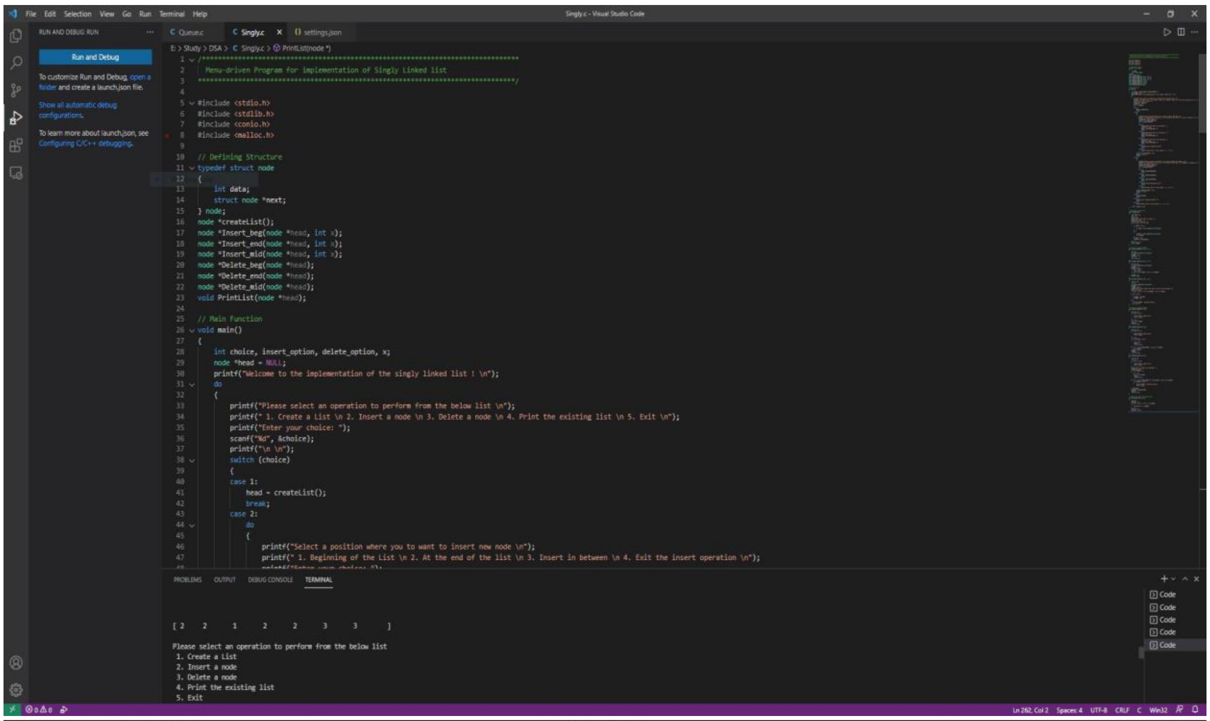
3) List of objects in a 3D game that need to be rendered to the screen.

★ Conclusion

→ Thus, we have studied the concepts and implementation of singly linked list and its various operation.

★ Outcome

→ Apply the concepts of singly, circular and doubly linked list for real-world application.



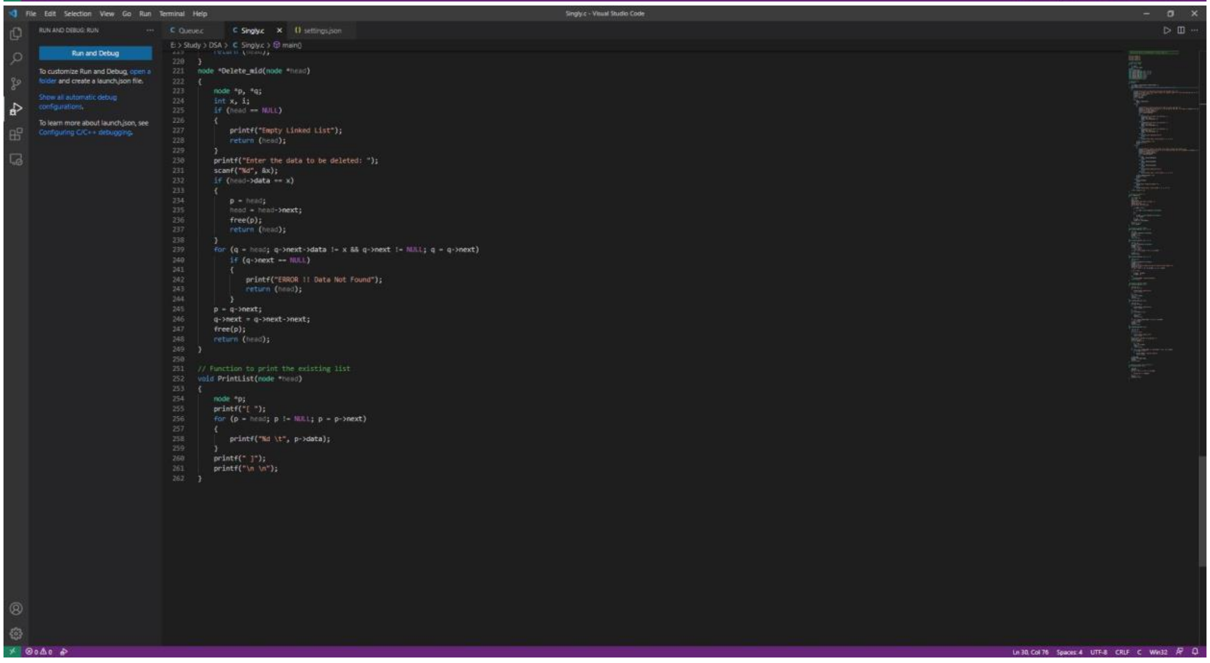
To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```c
/*********************************************************************************
  Menu-driven Program for implementation of Singly Linked list
*********************************************************************************/

#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");
                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");
```

```
[ 2    2     1     2     2     3     3      ]

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
```

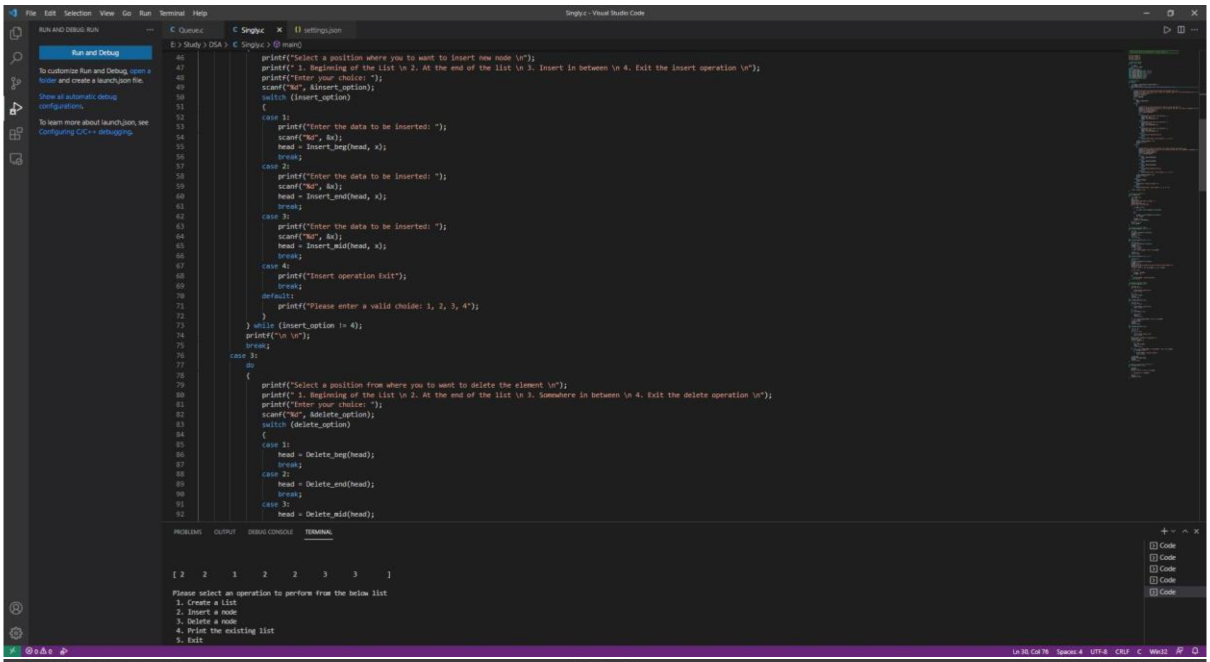


To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

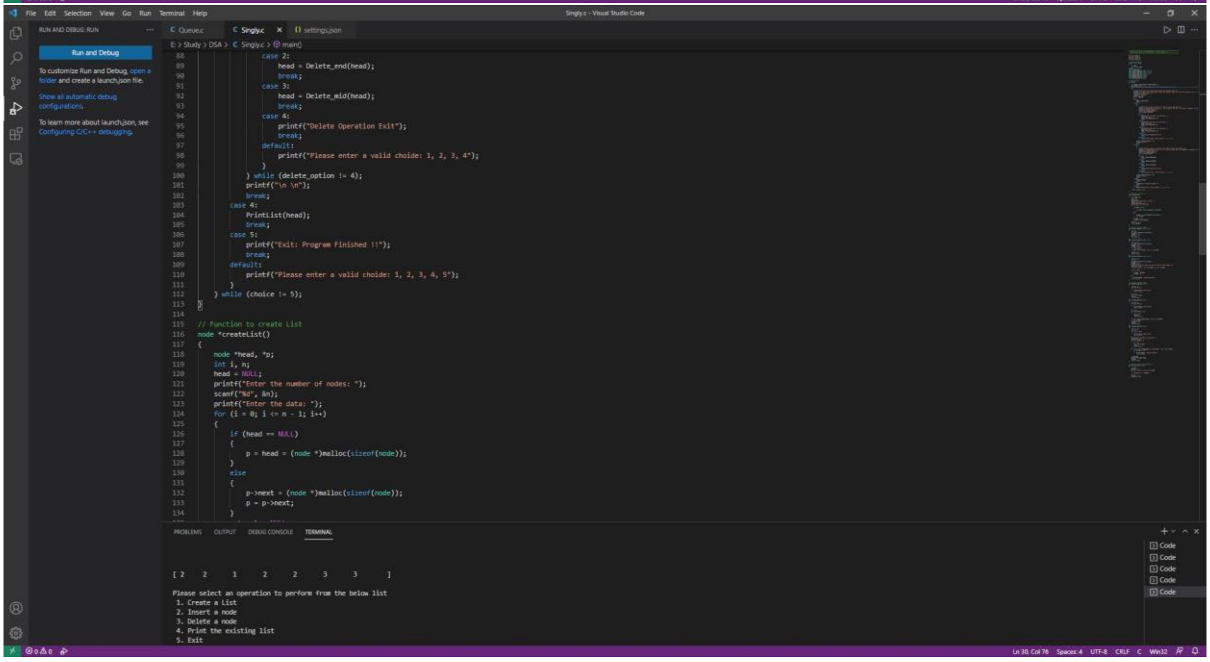
To learn more about launch.json, see Configuring C/C++ debugging.

```c
}
node *Delete_mid(node *head)
{
    node *p, *q;
    int x, i;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    printf("Enter the data to be deleted: ");
    scanf("%d", &x);
    if (head->data == x)
    {
        p = head;
        head = head->next;
        free(p);
        return (head);
    }
    for (q = head; q->next->data != x && q->next != NULL; q = q->next)
        if (q->next == NULL)
        {
            printf("ERROR !! Data Not Found");
            return (head);
        }
    p = q->next;
    q->next = q->next->next;
    free(p);
    return (head);
}

// Function to print the existing list
void PrintList(node *head)
{
    node *p;
    printf("[ ");
    for (p = head; p != NULL; p = p->next)
    {
        printf("%d \t", p->data);
    }
    printf(" ]");
    printf("\n \n");
}
```


File Edit Selection View Go Run Terminal Help
Singly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
Queue.c
Singly.c
settings.json
E: > Study > DSA > Singly.c > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
Ln 30, Col 76 Spaces: 4 UTF-8 CRLF C Win32

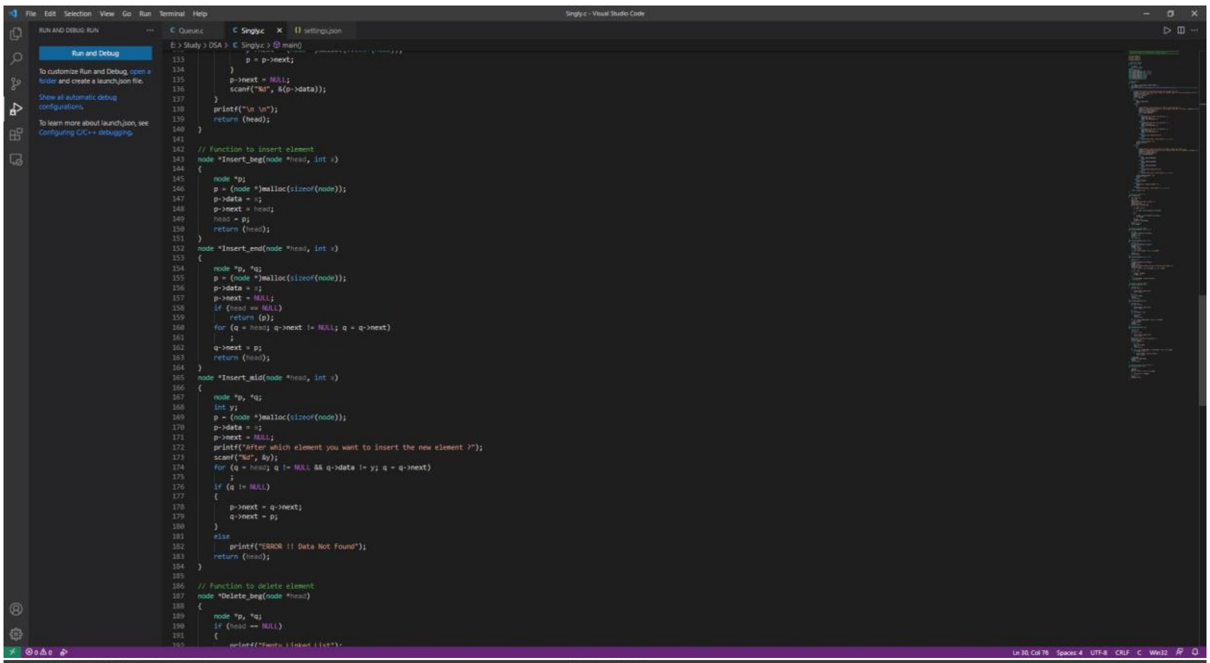
```c
                printf("Select a position where you to want to insert new node \n");
                printf(" 1. Beginning of the List \n 2. At the end of the list \n 3. Insert in between \n 4. Exit the insert operation \n");
                printf("Enter your choice: ");
                scanf("%d", &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:
                    head = Delete_beg(head);
                    break;
                case 2:
                    head = Delete_end(head);
                    break;
                case 3:
                    head = Delete_mid(head);
```

```
[ 2    2    1    2    2    3    3    ]

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
```

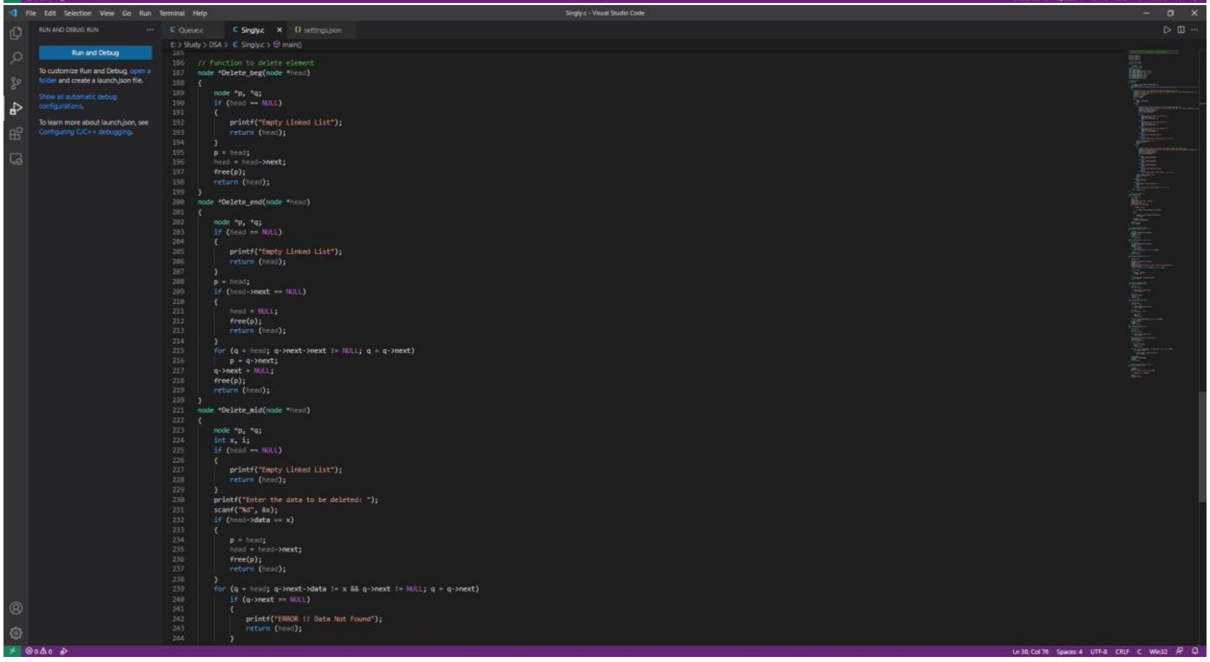

File Edit Selection View Go Run Terminal Help
Singly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
Queue.c
Singly.c
settings.json
E: > Study > DSA > Singly.c > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
Ln 30, Col 76 Spaces: 4 UTF-8 CRLF C Win32

```c
                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
node *createList()
{
    node *head, *p;
    int i, n;
    head = NULL;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the data: ");
    for (i = 0; i <= n - 1; i++)
    {
        if (head == NULL)
        {
            p = head = (node *)malloc(sizeof(node));
        }
        else
        {
            p->next = (node *)malloc(sizeof(node));
            p = p->next;
        }
```

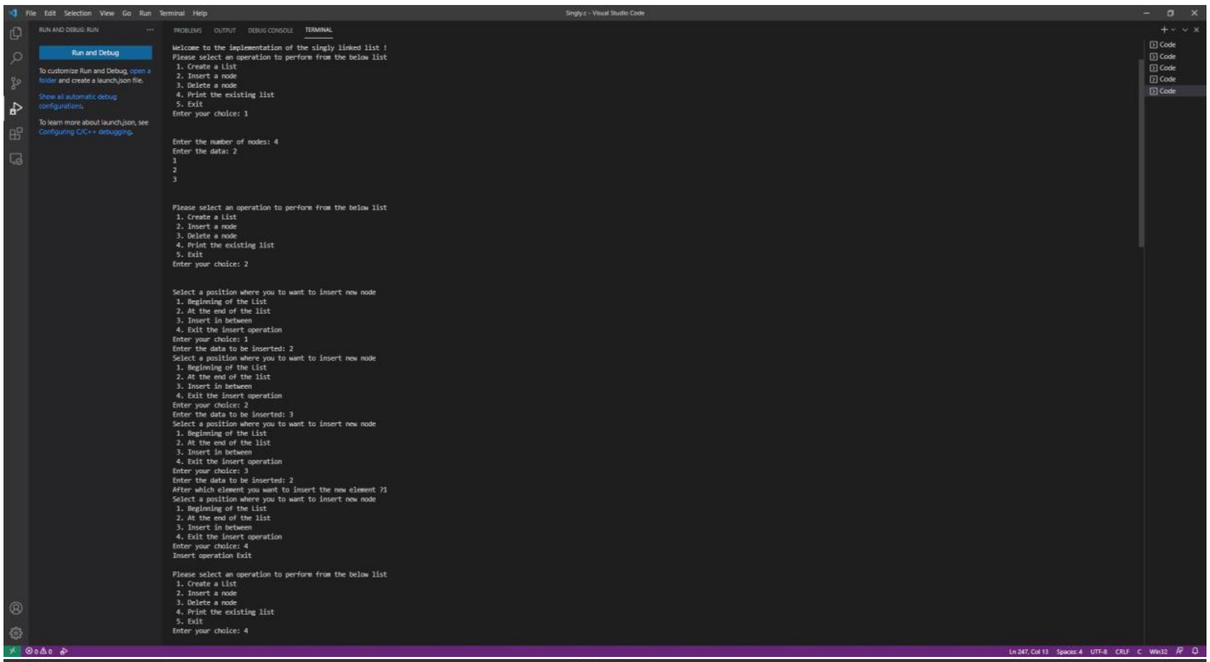
```
[ 2    2    1    2    2    3    3    ]

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
```


Singly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
Queue.c
Singly.c
settings.json
E: > Study > DSA > Singly.c > main()
Ln 30, Col 76 Spaces: 4 UTF-8 CRLF C Win32

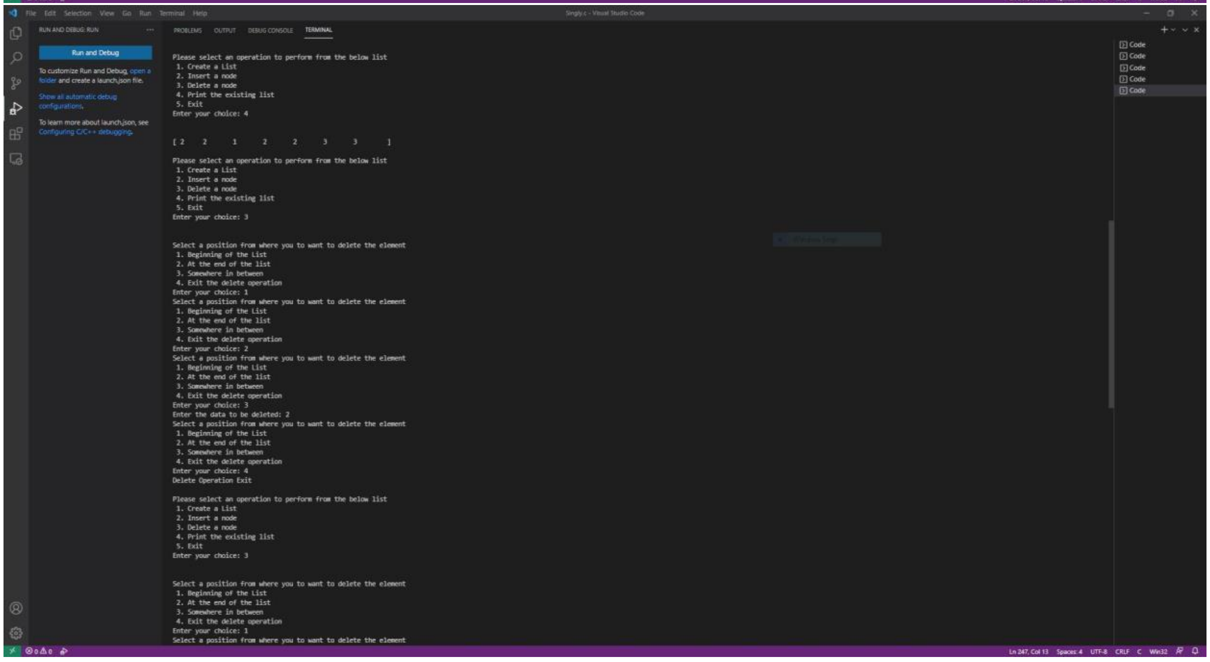
```c
                p = p->next;
            }
            p->next = NULL;
            scanf("%d", &(p->data));
        }
        printf("\n \n");
        return (head);
    }

    // Function to insert element
    node *Insert_beg(node *head, int x)
    {
        node *p;
        p = (node *)malloc(sizeof(node));
        p->data = x;
        p->next = head;
        head = p;
        return (head);
    }
    node *Insert_end(node *head, int x)
    {
        node *p, *q;
        p = (node *)malloc(sizeof(node));
        p->data = x;
        p->next = NULL;
        if (head == NULL)
            return (p);
        for (q = head; q->next != NULL; q = q->next)
            ;
        q->next = p;
        return (head);
    }
    node *Insert_mid(node *head, int x)
    {
        node *p, *q;
        int y;
        p = (node *)malloc(sizeof(node));
        p->data = x;
        p->next = NULL;
        printf("After which element you want to insert the new element ?");
        scanf("%d", &y);
        for (q = head; q != NULL && q->data != y; q = q->next)
            ;
        if (q != NULL)
        {
            p->next = q->next;
            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)
        {
```


Singly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
Queue.c
Singly.c
settings.json
E: > Study > DSA > Singly.c > main()
Ln 30, Col 76 Spaces: 4 UTF-8 CRLF C Win32

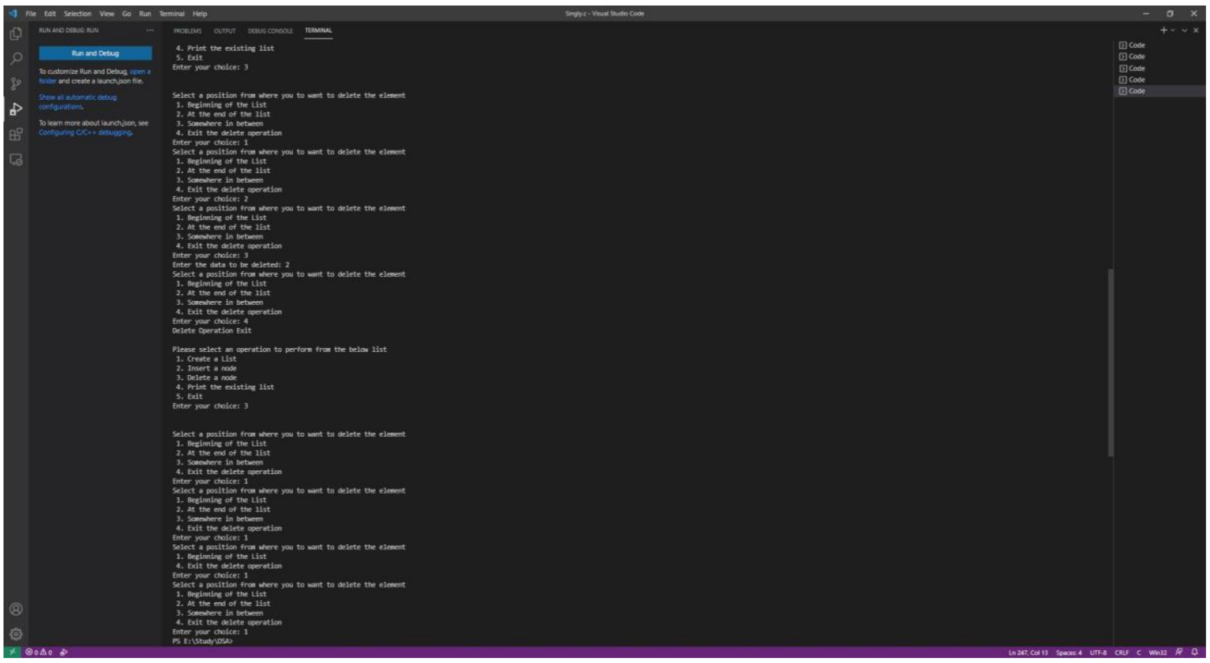
```c
    // Function to delete element
    node *Delete_beg(node *head)
    {
        node *p, *q;
        if (head == NULL)
        {
            printf("Empty Linked List");
            return (head);
        }
        p = head;
        head = head->next;
        free(p);
        return (head);
    }
    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);
    }
    node *Delete_mid(node *head)
    {
        node *p, *q;
        int x, i;
        if (head == NULL)
        {
            printf("Empty Linked List");
            return (head);
        }
        printf("Enter the data to be deleted: ");
        scanf("%d", &x);
        if (head->data == x)
        {
            p = head;
            head = head->next;
            free(p);
            return (head);
        }
        for (q = head; q->next->data != x && q->next != NULL; q = q->next)
            if (q->next == NULL)
            {
                printf("ERROR !! Data Not Found");
                return (head);
            }
```



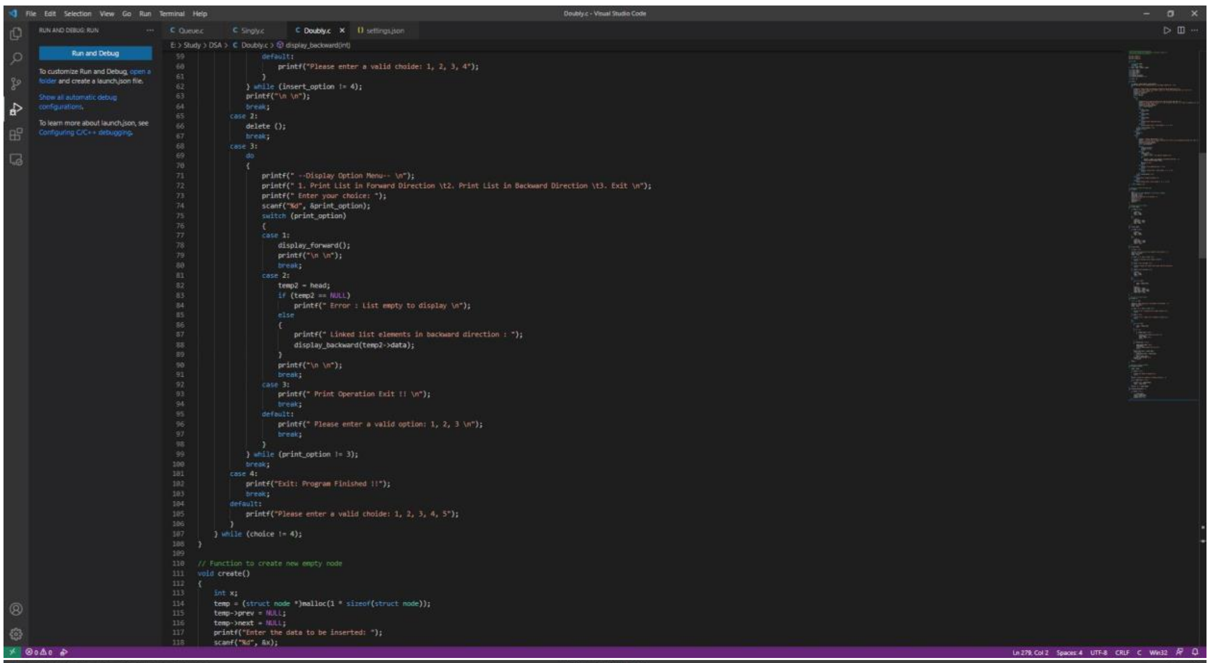
```
Welcome to the implementation of the singly linked list !
Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 1


Enter the number of nodes: 4
Enter the data: 2
1
2
3


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: 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: 2
Enter the data to be inserted: 3
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: 2
After which element you want to insert the new element ?1
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
```



```
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


[ 2    2      1      2      2      3      3       ]

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


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: 2
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: 2
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: 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
```



```
 4. Print the existing list
 5. Exit
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: 2
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: 2
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: 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: 1
Select a position from where you to want to delete the element
 1. Beginning of the List
 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: 1
PS E:\Study\DSA>
```



RUN AND DEBUG: RUN

To customize Run and Debug, open a folder and create a launch.json file.

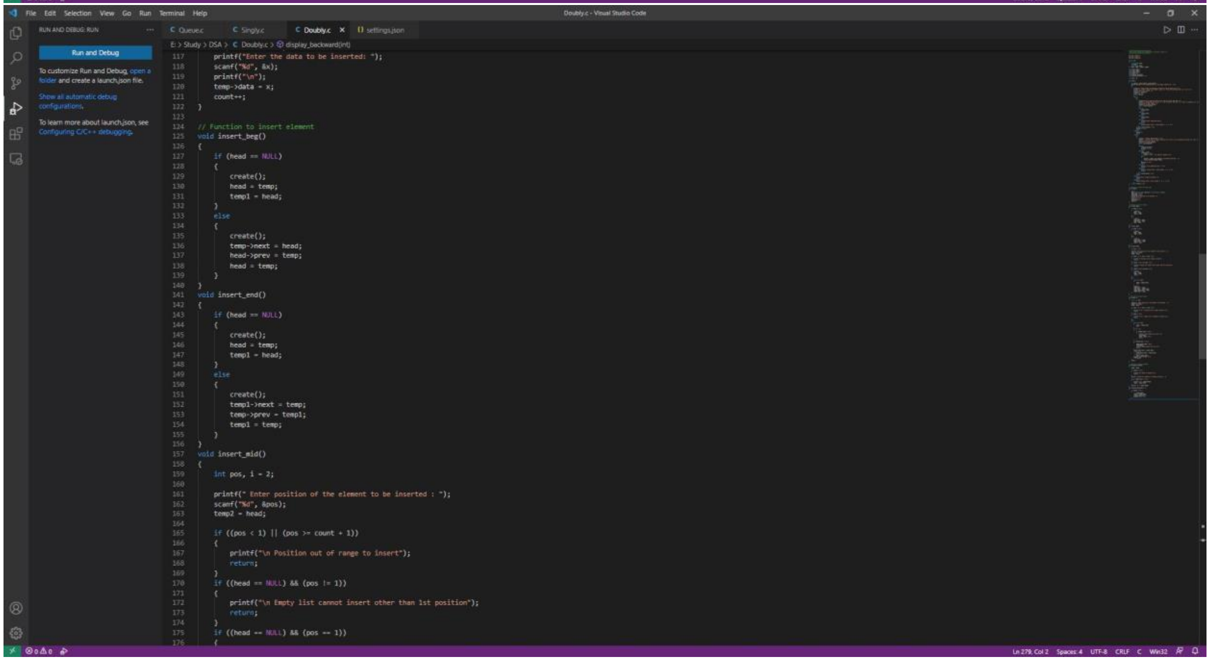
Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

E: > Study > DSA > Doubly.c > display_backward(int)

```c
                default:
                    printf("Please enter a valid choide: 1, 2, 3, 4");
                }
            } while (insert_option != 4);
            printf("\n \n");
            break;
        case 2:
            delete ();
            break;
        case 3:
            do
            {
                printf(" --Display Option Menu-- \n");
                printf(" 1. Print List in Forward Direction \t2. Print List in Backward Direction \t3. Exit \n");
                printf(" Enter your choice: ");
                scanf("%d", &print_option);
                switch (print_option)
                {
                case 1:
                    display_forward();
                    printf("\n \n");
                    break;
                case 2:
                    temp2 = head;
                    if (temp2 == NULL)
                        printf(" Error : List empty to display \n");
                    else
                    {
                        printf(" Linked list elements in backward direction : ");
                        display_backward(temp2->data);
                    }
                    printf("\n \n");
                    break;
                case 3:
                    printf(" Print Operation Exit !! \n");
                    break;
                default:
                    printf(" Please enter a valid option: 1, 2, 3 \n");
                    break;
                }
            } while (print_option != 3);
            break;
        case 4:
            printf("Exit: Program Finished !!");
            break;
        default:
            printf("Please enter a valid choide: 1, 2, 3, 4, 5");
        }
    } while (choice != 4);
}

// Function to create new empty node
void create()
{
    int x;
    temp = (struct node *)malloc(1 * sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("Enter the data to be inserted: ");
    scanf("%d", &x);
```

Ln 279, Col 2 Spaces: 4 UTF-8 CRLF C Win32



RUN AND DEBUG: RUN

To customize Run and Debug, open a folder and create a launch.json file.

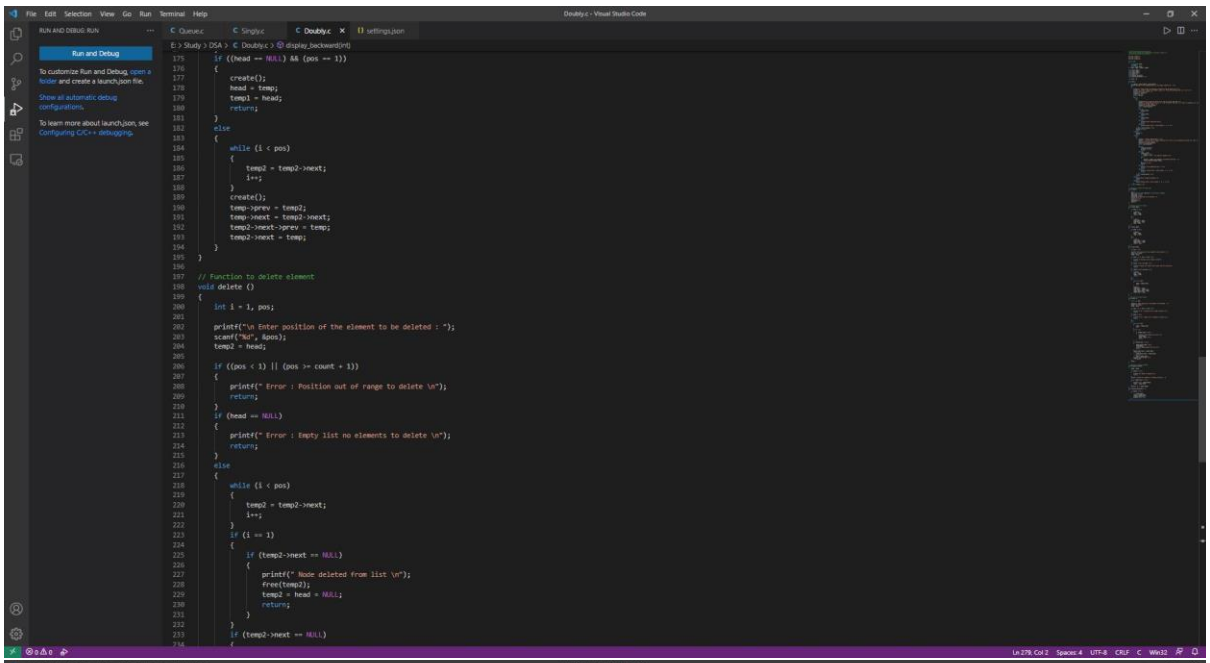
Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

E: > Study > DSA > Doubly.c > display_backward(int)

```c
    printf("Enter the data to be inserted: ");
    scanf("%d", &x);
    printf("\n");
    temp->data = x;
    count++;
}

// Function to insert element
void insert_beg()
{
    if (head == NULL)
    {
        create();
        head = temp;
        temp1 = head;
    }
    else
    {
        create();
        temp->next = head;
        head->prev = temp;
        head = temp;
    }
}
void insert_end()
{
    if (head == NULL)
    {
        create();
        head = temp;
        temp1 = head;
    }
    else
    {
        create();
        temp1->next = temp;
        temp->prev = temp1;
        temp1 = temp;
    }
}
void insert_mid()
{
    int pos, i = 2;

    printf(" Enter position of the element to be inserted : ");
    scanf("%d", &pos);
    temp2 = head;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Position out of range to insert");
        return;
    }
    if ((head == NULL) && (pos != 1))
    {
        printf("\n Empty list cannot insert other than 1st position");
        return;
    }
    if ((head == NULL) && (pos == 1))
    {
```

Ln 279, Col 2 Spaces: 4 UTF-8 CRLF C Win32

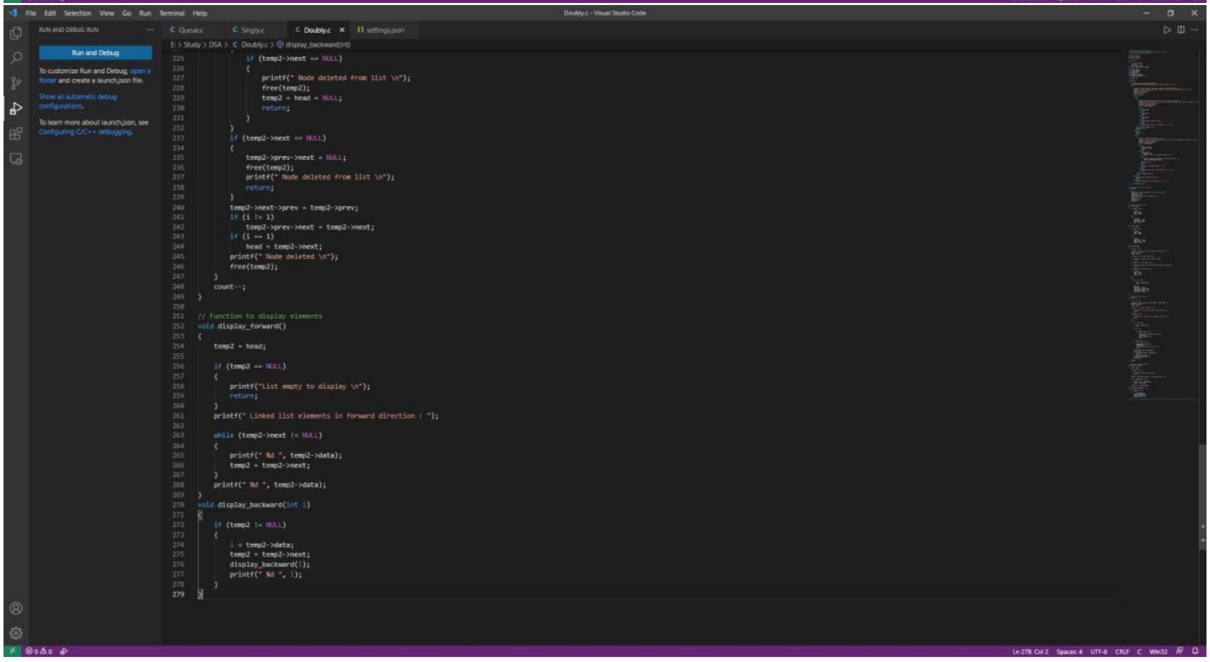

Doubly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
Queue.c
Singly.c
Doubly.c
settings.json
E: > Study > DSA > Doubly.c > display_backward(int)
Ln 279, Col 2   Spaces: 4   UTF-8   CRLF   C   Win32

To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```c
    if ((head == NULL) && (pos == 1))
    {
        create();
        head = temp;
        temp1 = head;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        create();
        temp->prev = temp2;
        temp->next = temp2->next;
        temp2->next->prev = temp;
        temp2->next = temp;
    }
}

// Function to delete element
void delete ()
{
    int i = 1, pos;

    printf("\n Enter position of the element to be deleted : ");
    scanf("%d", &pos);
    temp2 = head;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf(" Error : Position out of range to delete \n");
        return;
    }
    if (head == NULL)
    {
        printf(" Error : Empty list no elements to delete \n");
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf(" Node deleted from list \n");
                free(temp2);
                temp2 = head = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
```

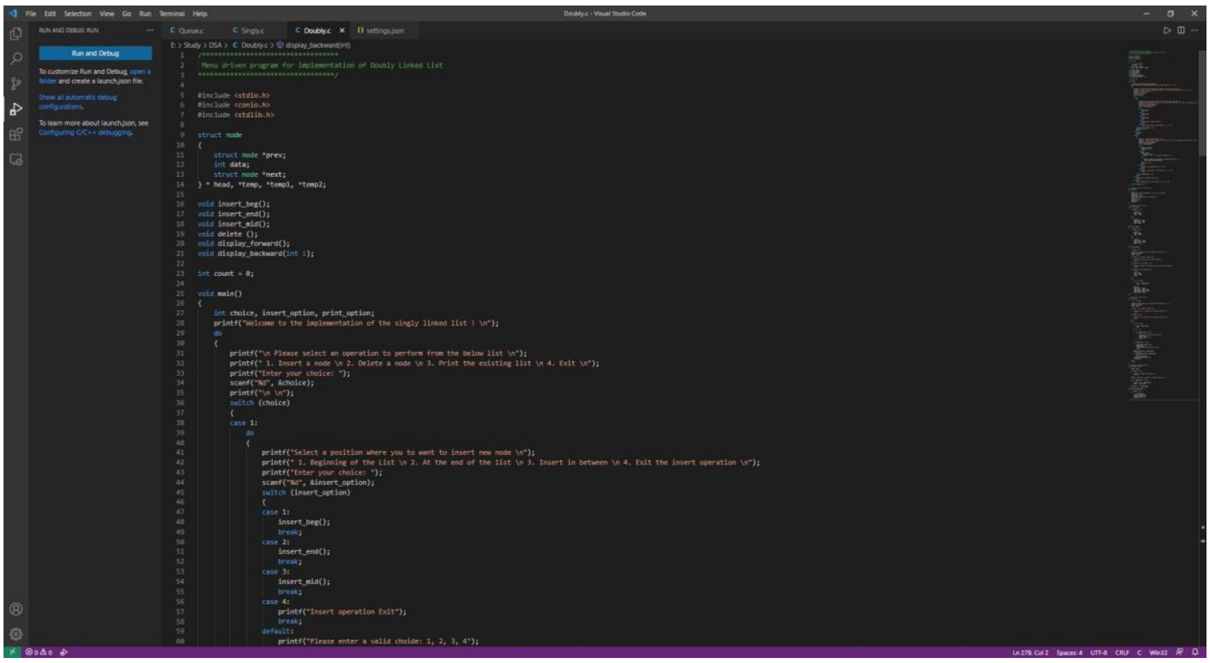

Doubly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
Queue.c
Singly.c
Doubly.c
settings.json
E: > Study > DSA > Doubly.c > display_backward(int)
Ln 279, Col 2   Spaces: 4   UTF-8   CRLF   C   Win32

To customize Run and Debug, open a folder and create a launch.json file.

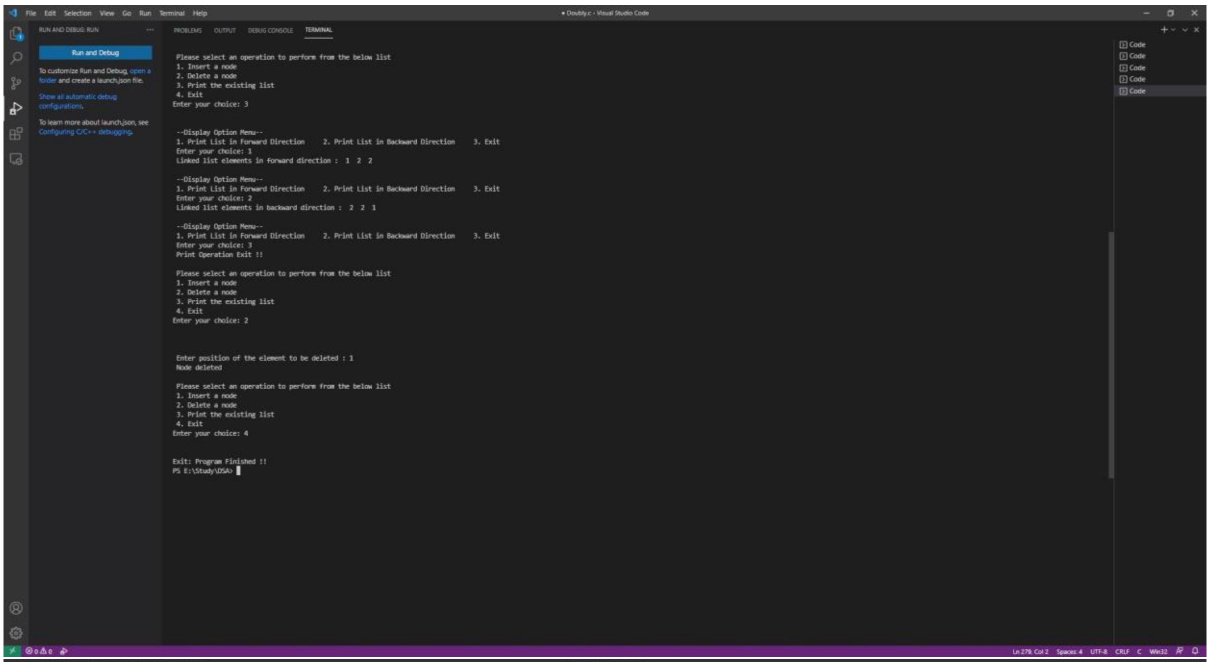
Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

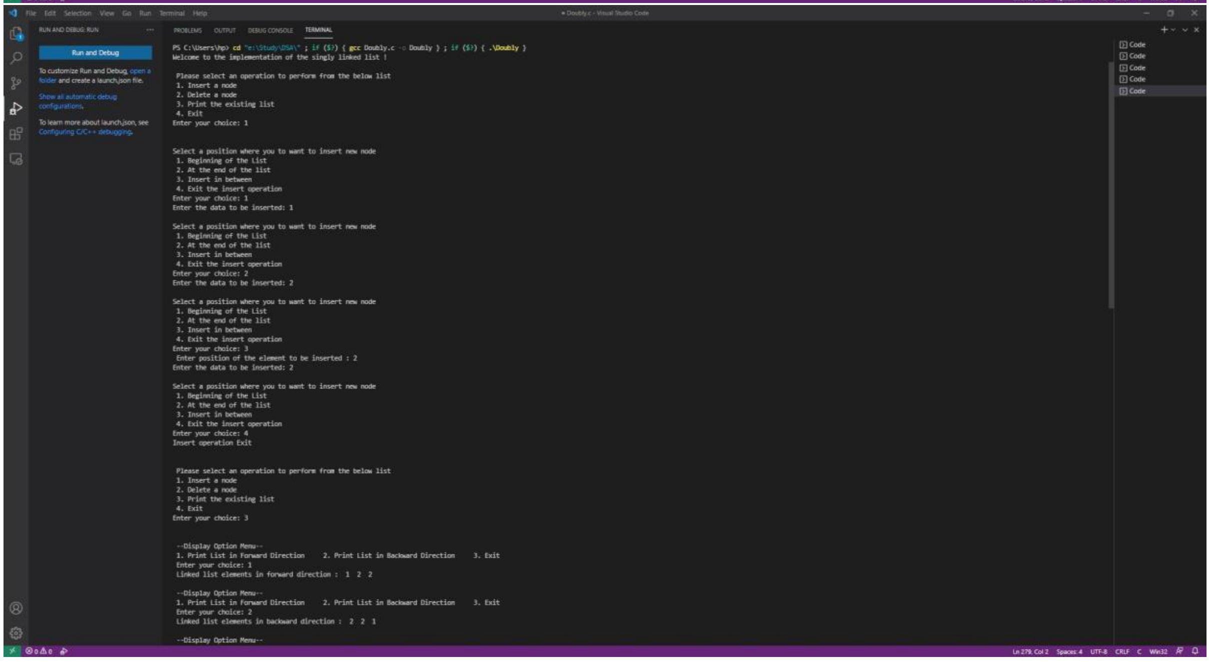
```c
            if (temp2->next == NULL)
            {
                printf(" Node deleted from list \n");
                free(temp2);
                temp2 = head = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf(" Node deleted from list \n");
            return;
        }
        temp2->next->prev = temp2->prev;
        if (i != 1)
            temp2->prev->next = temp2->next;
        if (i == 1)
            head = temp2->next;
        printf(" Node deleted \n");
        free(temp2);
    }
    count--;
}

// Function to display elements
void display_forward()
{
    temp2 = head;

    if (temp2 == NULL)
    {
        printf("List empty to display \n");
        return;
    }
    printf(" Linked list elements in forward direction : ");

    while (temp2->next != NULL)
    {
        printf(" %d ", temp2->data);
        temp2 = temp2->next;
    }
    printf(" %d ", temp2->data);
}
void display_backward(int i)
{
    if (temp2 != NULL)
    {
        i = temp2->data;
        temp2 = temp2->next;
        display_backward(i);
        printf(" %d ", i);
    }
}
```


File Edit Selection View Go Run Terminal Help
Doubly.c - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
Queue.c
Singly.c
Doubly.c
settings.json
E: > Study > DSA > Doubly.c > display_backward(int)
Ln 279, Col 2 Spaces: 4 UTF-8 CRLF C Win32

```c
/*********************************
 Menu driven program for implementation of Doubly Linked List
*********************************/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct node
{
    struct node *prev;
    int data;
    struct node *next;
} * head, *temp, *temp1, *temp2;

void insert_beg();
void insert_end();
void insert_mid();
void delete ();
void display_forward();
void display_backward(int i);

int count = 0;

void main()
{
    int choice, insert_option, print_option;
    printf("Welcome to the implementation of the singly linked list ! \n");
    do
    {
        printf("\n Please select an operation to perform from the below list \n");
        printf(" 1. Insert a node \n 2. Delete a node \n 3. Print the existing list \n 4. Exit \n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        printf("\n \n");
        switch (choice)
        {
        case 1:
            do
            {
                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:
                    insert_beg();
                    break;
                case 2:
                    insert_end();
                    break;
                case 3:
                    insert_mid();
                    break;
                case 4:
                    printf("Insert operation Exit");
                    break;
                default:
                    printf("Please enter a valid choide: 1, 2, 3, 4");
```



```
 Please select an operation to perform from the below list
 1. Insert a node
 2. Delete a node
 3. Print the existing list
 4. Exit
Enter your choice: 3


 --Display Option Menu--
 1. Print List in Forward Direction     2. Print List in Backward Direction     3. Exit
 Enter your choice: 1
 Linked list elements in forward direction :  1  2  2

 --Display Option Menu--
 1. Print List in Forward Direction     2. Print List in Backward Direction     3. Exit
 Enter your choice: 2
 Linked list elements in backward direction :  2  2  1

 --Display Option Menu--
 1. Print List in Forward Direction     2. Print List in Backward Direction     3. Exit
 Enter your choice: 3
 Print Operation Exit !!

 Please select an operation to perform from the below list
 1. Insert a node
 2. Delete a node
 3. Print the existing list
 4. Exit
Enter your choice: 2



 Enter position of the element to be deleted : 1
 Node deleted

 Please select an operation to perform from the below list
 1. Insert a node
 2. Delete a node
 3. Print the existing list
 4. Exit
Enter your choice: 4


Exit: Program Finished !!
PS E:\Study\DSA> 
```



```
PS C:\Users\hp> cd "e:\Study\DSA\" ; if ($?) { gcc Doubly.c -o Doubly } ; if ($?) { .\Doubly }
Welcome to the implementation of the singly linked list !

 Please select an operation to perform from the below list
 1. Insert a node
 2. Delete a node
 3. Print the existing list
 4. Exit
Enter your choice: 1


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: 1

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: 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: 3
 Enter position of the element to be inserted : 2
Enter the data to be inserted: 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: 4
Insert operation Exit


 Please select an operation to perform from the below list
 1. Insert a node
 2. Delete a node
 3. Print the existing list
 4. Exit
Enter your choice: 3


 --Display Option Menu--
 1. Print List in Forward Direction     2. Print List in Backward Direction     3. Exit
 Enter your choice: 1
 Linked list elements in forward direction :  1  2  2

 --Display Option Menu--
 1. Print List in Forward Direction     2. Print List in Backward Direction     3. Exit
 Enter your choice: 2
 Linked list elements in backward direction :  2  2  1

 --Display Option Menu--
```