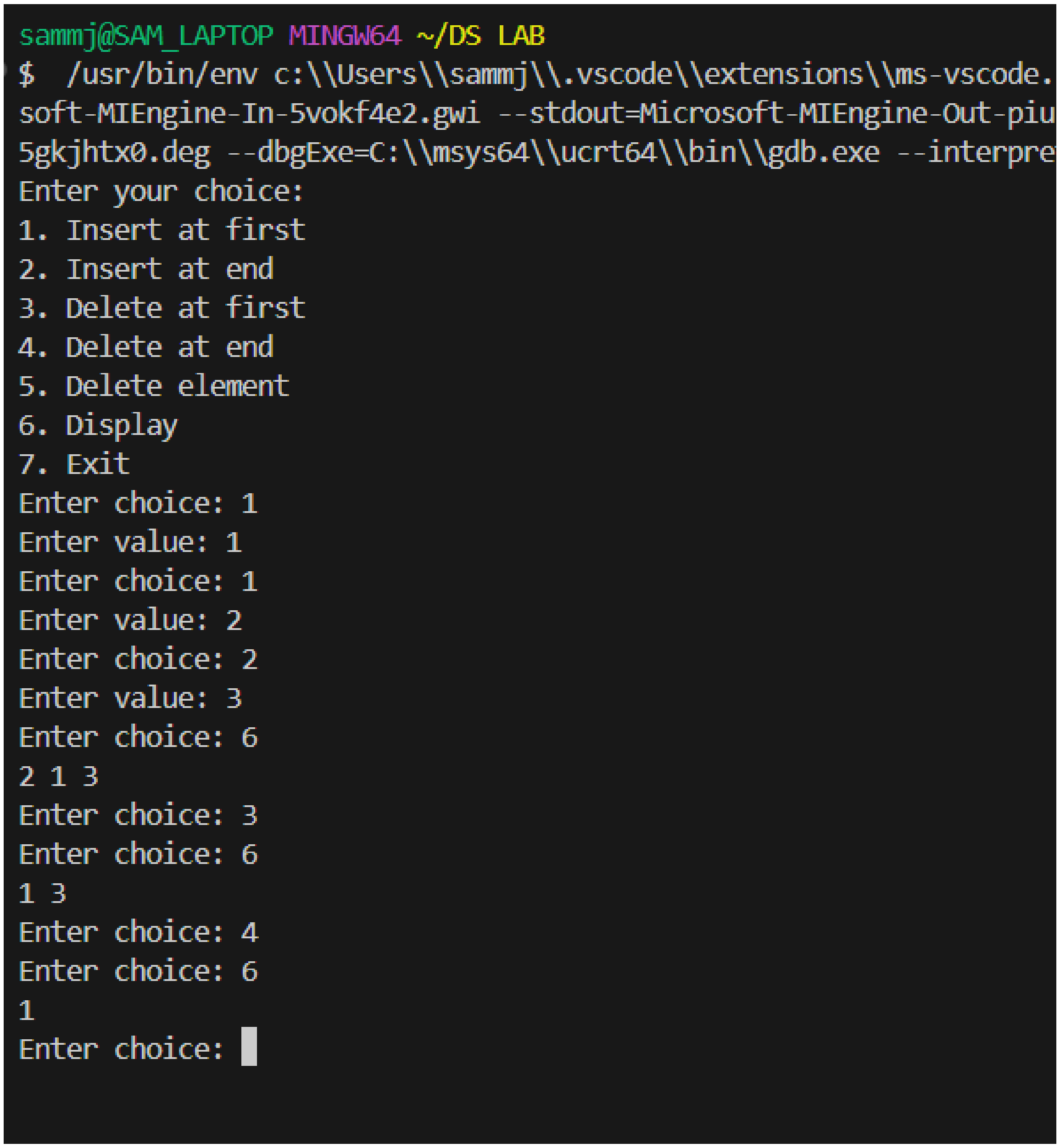
```c
// Implementation of a linked list in C
#include <stdio.h>
#include <stdlib.h>

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void create(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void insertFirst(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}

void deleteFirst(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty");
    } else {
        struct Node* temp = *head;
        *head = (*head)->next;
        free(temp);
    }
}
```

```c
void deleteLast(struct Node** head) {
    if (*head == NULL) {
        printf("List is empty");
    } else {
        if ((*head)->next == NULL) {
            free(*head);
            *head = NULL;
        } else {
            struct Node* temp = *head;
            while (temp->next && temp->next->next != NULL) {
                temp = temp->next;
            }
            free(temp->next);
            temp->next = NULL;
        }
    }
}

void deleteElement(struct Node** head, int key) {
    if (*head == NULL) {
        printf("List is empty");
    } else {
        struct Node* temp = *head;
        if (temp != NULL && temp->data == key) {
            *head = temp->next;
            free(temp);
        } else {
            struct Node* prev = NULL;
            while (temp != NULL && temp->data != key) {
                prev = temp;
                temp = temp->next;
            }
            if (temp == NULL) {
                printf("Element not found");
            } else {
                prev->next = temp->next;
                free(temp);
            }
        }
    }
}

void display(struct Node* head) {
    if (head == NULL) {
        printf("List is empty");
    } else {
        struct Node* temp = head;
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}

int main() {
    struct Node* head = NULL;
    int choice, value;

    printf("Enter your choice:\n1. Insert at first\n2. Insert at end\n3. Delete
at first\n4. Delete at end\n5. Delete element\n6. Display\n7. Exit\n");

    while (1) {
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &value);
                insertFirst(&head, value);
                break;
            case 2:
                printf("Enter value: ");
                scanf("%d", &value);
                create(&head, value);
                break;
            case 3:
                deleteFirst(&head);
                break;
            case 4:
                deleteLast(&head);
                break;
            case 5:
                printf("Enter element to delete: ");
                scanf("%d", &value);
                deleteElement(&head, value);
                break;
            case 6:
                display(head);
                break;
            case 7:
            default:
                return 0;
        }
    }

    return 0;
}
```

Output:



```
sammj@SAM_LAPTOP MINGW64 ~/DS LAB
$  /usr/bin/env c:\\Users\\sammj\\.vscode\\extensions\\ms-vscode.
soft-MIEngine-In-5vokf4e2.gwi --stdout=Microsoft-MIEngine-Out-piu
5gkjhtx0.deg --dbgExe=C:\\msys64\\ucrt64\\bin\\gdb.exe --interpre
Enter your choice:
1. Insert at first
2. Insert at end
3. Delete at first
4. Delete at end
5. Delete element
6. Display
7. Exit
Enter choice: 1
Enter value: 1
Enter choice: 1
Enter value: 2
Enter choice: 2
Enter value: 3
Enter choice: 6
2 1 3
Enter choice: 3
Enter choice: 6
1 3
Enter choice: 4
Enter choice: 6
1
Enter choice: 
```

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

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

struct Node* createNode(int data)
{
   struct Node* newNode = (struct Node*) malloc
                                   (sizeof (struct Node));
   newNode → data = data;
   newNode → next = NULL;
   return newNode;
}

~~void create(struct Node** head, int data)~~
~~{~~
~~  struct Node* newnode = createNode(data);~~
~~  if (*head = = NULL)~~
~~      *head = newNode;~~
~~  else~~
~~     struct Node* temp = *head;~~
~~     while (temp → next! = NULL~~

void insert~~first~~end (struct Node** head, int data)
{
   struct Node* newnode = createNode (data);
   if (* head == NULL)
        * head = newNode;
   else
```

```
        {
            struct Node* temp = *head
            while ( temp -> next! = NULL)
            {
                temp = temp -> next;
            }
            temp -> next = newNode;
        }
}
void insertFirst ( struct Node **head, int data)
 {
    struct Node* newNode = createNode(data)
    ~~*head = newNode;~~
    ~~printf ("Node = d~~
    newnode -> next = *head
     * head = newNode
}
void deletefirst ( struct Node** head)
{
    if (*head == NULL)
     {
        printf(" List is empty");
     }
    else
    {
       struct Node* temp = *head;
       ~~free(temp)~~ *head = (*head) -> next;
        free (temp);
    }
}
```

PAGE :
DATE : / /

```
void deletelast (struct Node**head)
{
    if (*head = =NULL)
     {
        printf("List is empty");
     }
    else
     {
        if((*head) → next = =NULL)
         {
            free (*head);
            *head = NULL;
         }
         else
         {
            struct Node* temp = *head;
            while ( temp → next && temp → next → next != NULL)
             {
                temp = temp → next;
             }
             free (temp → next);
             temp → next = NULL;

         }
    }
}
```

```
void deleteElement (struct Node**head{, int key)
{
    if (*head = = NULL)
    {
        printf(" list is empty ");
    }
    else
    {
        struct Node* temp = * head;
        if ( temp != NULL && temp -> data = =key)
        {
            * head = temp -> next ;
            free(temp);
        }
        else
        {
            struct Node* prev = NULL;
            while (temp != NULL && temp -> data != key)
            {
                prev = temp;
                temp = temp -> next;
            }
            if (temp = = NULL)
            {
                printf(" Element not found");
            }
            else
            {  prev -> next = temp -> next
                free(temp);
            }
        }
    }
}
```

PAGE :
DATE :

```
void display (struct Node ** head)
 {
      if (*head = = NULL)
      {
          printf ("List empty")
      }
      else
      {
          struct Node* temp = head;
          while (temp ! = NULL)
          {
              printf ("%d", temp -> data);
              temp = temp -> next;
          }
      }
}
void main ()
 {
   struct Node* head = NULL;
   int choice, value;
   printf ("Enter 1. Insert at First | 2. Insert at End | 3. Delete at First |
          4. delete at End  5 - Delete Element");
   scanf ("%d", &choice);
while(1)
 { switch (choice)
   {
     case 1:
          printf ("Enter value : ")
          scanf ("%d, &value");
          ~~break~~ ;
          insertFirst (&head, value);
          break;
```

PAGE :
DATE : / /

```
case 2:
        printf("Enter value:);
        scanf("%d", &value);
        insert~~At~~End (&head, value);
        break;

case 3:
        deleteFirst (&head);
        break;
case 4:
        deleteLast (&head);
        break;

case 5:
        printf("Enter element to delete");
        scanf("%d", & value);
        deleteElement (&head, value);
        break;
case 6:
        display (&head);
        break;

~~case 7:~~
default:
        ~~break;~~ return 0;
    }
  }
}
```

Seen

o/p Seen ~~Input~~ Op:

```
1. Insert at First 2. - - -        : 1
   Enter value: 1
1 Enter
```