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

struct node {
int info;
struct node* link;
};
struct node* start = NULL;


void createList()
{
    if (start == NULL) {
        int n;
        printf("\nEnter the number of nodes: ");
        scanf("%d", &n);

        if (n > 0) {
            int data;
            struct node* newnode;
            struct node* temp;

            newnode = malloc(sizeof(struct node));
            start = newnode;
            temp = start;

            printf("\nEnter number to be inserted : ");
            scanf("%d", &data);
            start->info = data;

            for (int i = 2; i <= n; i++) {
                newnode = malloc(sizeof(struct node));
                temp->link = newnode;

                printf("\nEnter number to be inserted : ");
                scanf("%d", &data);
                newnode->info = data;
```

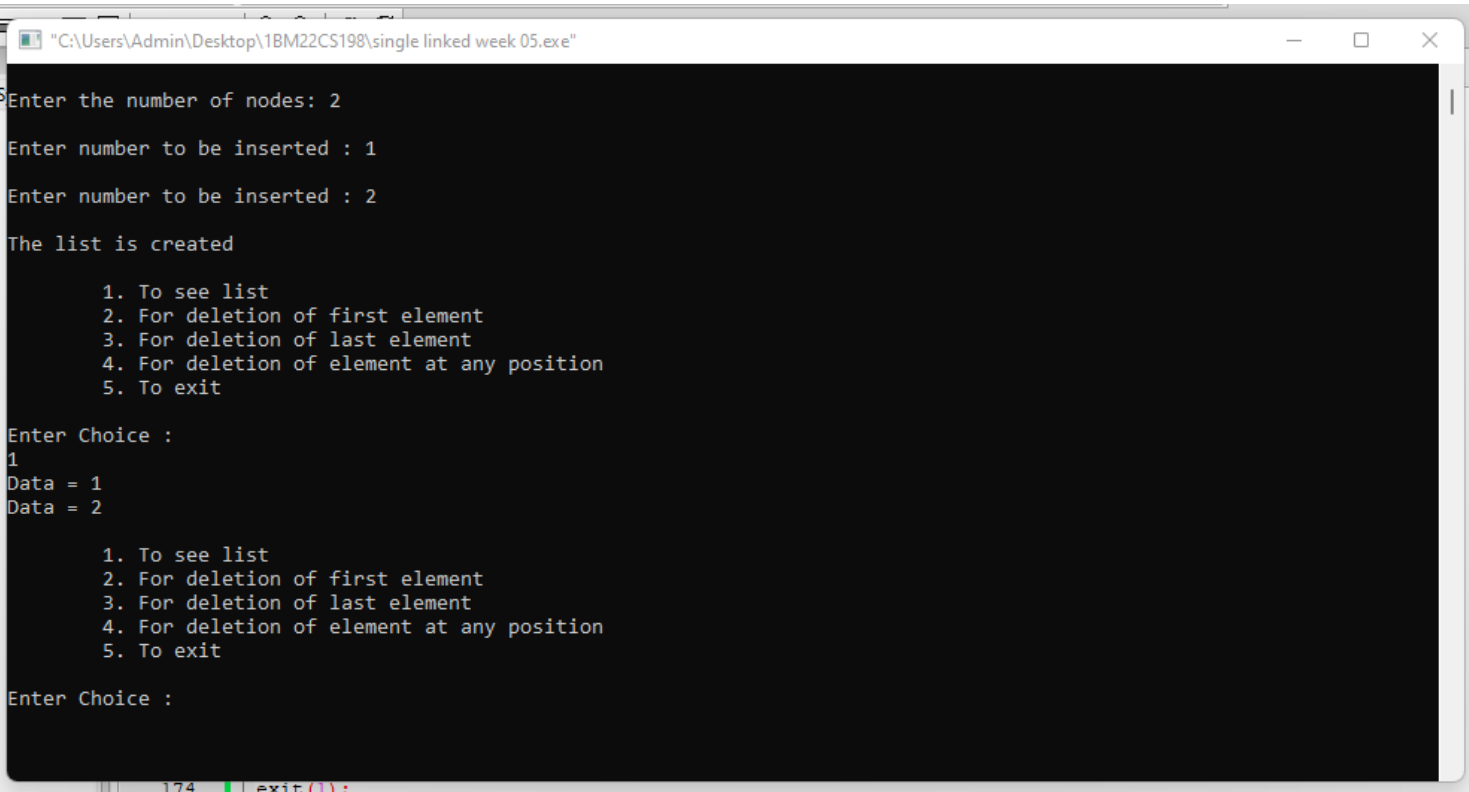
```
                newnode->info = data;
                temp = temp->link;
            }

            temp->link = NULL;
        }

        printf("\nThe list is created\n");
    } else {
        printf("\nThe list is already created\n");
    }
}



void display()
{
struct node* temp;


if (start == NULL)
printf("\nList is empty\n");


else {
temp = start;
while (temp != NULL) {
printf("Data = %d\n", temp->info);
temp = temp->link;
}
}
}


void deleteFirst()
{
struct node* temp;
```

```c
struct node* temp;
if (start == NULL)
printf("\nList is empty\n");
else {
temp = start;
start = start->link;
free(temp); // Free the memory of the deleted node
}
}

void deleteEnd()
{
struct node *temp, *prevnode;
if (start == NULL)
printf("\nList is Empty\n");
else {
temp = start;
while (temp->link != NULL) {
prevnode = temp;
temp = temp->link;
}
free(temp); // Free the memory of the deleted node
prevnode->link = NULL;
}
}

void deletePosition()
{
    struct node *temp, *position, *prevnode;
    int i = 1, pos;

    if (start == NULL)
        printf("\nList is empty\n");
    else {
        printf("\nEnter index : ");
        scanf("%d", &pos);
```

```c
struct node *temp, *position, *prevnode;
int i = 1, pos;

if (start == NULL)
    printf("\nList is empty\n");
else {
    printf("\nEnter index : ");
    scanf("%d", &pos);

    if (pos <= 0) {
        printf("\nInvalid position\n");
        return;
    }

    temp = start;
    position = NULL;

    if (pos == 1) {
        start = start->link;
        free(temp); // Free the memory of the deleted node
        return;
    }

    while (i < pos && temp != NULL) {
        prevnode = temp;
        temp = temp->link;
        i++;
    }

    if (temp == NULL) {
        printf("\nInvalid position\n");
        return;
    }

    position = temp;
    prevnode->link = temp->link;
```

```
        prevnode->link = temp->link;
        free(position); // Free the memory of the deleted node
    }
}



int main()
{
createList();
int choice;
while (1) {

printf("\n\t1. To see list\n");
printf("\t2. For deletion of "
"first element\n");
printf("\t3. For deletion of "
"last element\n");
printf("\t4. For deletion of "
"element at any position\n");
printf("\t5. To exit\n");
printf("\nEnter Choice :\n");
scanf("%d", &choice);

switch (choice) {
case 1:
display();
break;
case 2:
deleteFirst();
break;
case 3:
deleteEnd();
break;
case 4:
deletePosition();
break;
```

```
switch (choice) {
case 1:
display();
break;
case 2:
deleteFirst();
break;
case 3:
deleteEnd();
break;
case 4:
deletePosition();
break;
case 5:
exit(1);
break;
default:
printf("Incorrect Choice\n");
}
}
return 0;
}
```




```
174     exit(1);
175     break;
176     default:
177     printf("Incorrect Choice\n");
```