1. **Write a menu driven program in C to perform Single Linear Linked List operations using structure pointer. (Create, Display, Count, Insertion , Deletion, Sort, Reverse).**

**Program: prg4.c**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

} \*head;

void create(int);

void display();

int count();

void insert();

void deletion(int item);

void search(int);

void sort();

void insertAtPosition(int, int);

void reverse();

int main()

{

int ch, item, num;

while (1)

{

printf("\n\t----- SINGLE LINKED LIST -------\n");

printf(" 0. Exit\n 1. Create\n 2. Display\n 3. Count\n 4. Insertion\n 5. Deletion\n 6. Search\n 7. Sort\n 8. Reverse\n ");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 0:

exit(0);

case 1:

printf("\nHow many nodes you want to create : ");

scanf("%d", &num);

create(num);

display();

break;

case 2:

display();

break;

case 3:

printf("\nThere are %d nodes in the Linked List\n", count());

break;

case 4:

display();

insert();

display();

break;

case 5:

printf("Enter the element you want to delete: ");

scanf("%d", &item);

display();

deletion(item);

display();

break;

case 6:

printf("\nEnter the item you want to search: ");

scanf("%d", &item);

display();

search(item);

break;

case 7:

printf("\nBefore Sorting ");

display();

sort();

printf("\nAfter Sorting ");

display();

break;

case 8:

reverse();

printf("\n\nLinked List reversed successfully.\n");

display();

break;

default:

printf("\n\t!!! Enter a correct choice !!!\t");

}

printf("\n\nPress Enter to continue...");

fflush(stdin);

getchar();

}

return 0;

}

void create(int size)

{

struct node \*newnode, \*temp;

int i, data;

head = (struct node \*)malloc(sizeof(struct node));

printf("Enter element 1 : ");

scanf("%d", &data);

head->data = data;

head->next = NULL;

temp = head;

for (i = 2; i <= size; i++)

{

newnode = (struct node \*)malloc(sizeof(struct node));

printf("Enter element %d : ", i);

scanf("%d", &data);

newnode->data = data;

newnode->next = NULL;

temp->next = newnode;

temp = temp->next;

}

}

void display()

{

struct node \*temp;

temp = head;

printf("\nThe Linked List is :\nHead -> ");

while (temp != NULL)

{

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL");

}

int count()

{

struct node \*temp;

int count = 0;

temp = head;

while (temp != NULL)

{

count++;

temp = temp->next;

}

return count;

}

void search(int item)

{

struct node \*temp;

int count = 1;

temp = head;

while (temp != NULL)

{

if (temp->data == item)

{

printf("\n%d found at position %d\n", item, count);

return;

}

count++;

temp = temp->next;

}

printf("\n%d not found in the Linked List\n", item);

}

void insert()

{

int option, data, position;

printf("\n\n---- INSERTION OPTIONS ----\n");

printf("\n1. Insertion at the beginning \n2. Insertion at the End \n3. Insertion before a node \n4. Insertion after a node \n5. Insertion at a given position");

printf("\nEnter your choice: ");

scanf("%d", &option);

printf("Enter your data: ");

scanf("%d", &data);

switch (option)

{

case 1:

insertAtPosition(data, 0);

printf("\nsuccessfully entered at the beginning: \n");

break;

case 2:

insertAtPosition(data, count() + 1);

printf("\nsuccessfully entered at the end: \n");

break;

case 3:

printf("enter the position: ", position);

scanf("%d", &position);

insertAtPosition(data, position - 1);

printf("\nsuccessfully entered before a node: \n");

break;

case 4:

printf("enter the position: ", position);

scanf("%d", &position);

insertAtPosition(data, position + 1);

printf("\nsuccessfully entered after a node: \n");

break;

case 5:

printf("enter the position: ", position);

scanf("%d", &position);

insertAtPosition(data, position);

printf("\nsuccessfully entered at position %d: \n", position);

break;

}

}

void insertAtPosition(int data, int position)

{

int count;

struct node \*temp, \*q;

temp = (struct node \*)malloc(sizeof(struct node));

temp->data = data;

temp->next = NULL;

if (position == 0)

{

temp->next = head;

head = temp;

}

else

{

q = head;

for (count = 1; count < position - 1; count++)

{

q = q->next;

}

temp->next = q->next;

q->next = temp;

}

}

void deletion(int data)

{

struct node \*temp, \*prev;

temp = head;

if (temp != NULL && temp->data == data)

{

head = temp->next;

free(temp);

return;

}

while (temp != NULL && temp->data != data)

{

prev = temp;

temp = temp->next;

}

if (temp == NULL)

{

printf("%d not found in the Linked List\n", data);

return;

}

prev->next = temp->next;

free(temp);

printf("\t%d deleted from the Linked List\n", data);

}

void reverse()

{

struct node \*prev = NULL;

struct node \*current = head;

struct node \*next;

while (current != NULL)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

void sort()

{

struct node \*temp1, \*temp2;

int tempData;

temp1 = head;

while (temp1->next != NULL)

{

temp2 = temp1->next;

while (temp2 != NULL)

{

if (temp1->data > temp2->data)

{

tempData = temp1->data;

temp1->data = temp2->data;

temp2->data = tempData;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

**OUTPUT:**

There are 3 nodes in the Linked List

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 4

The Linked List is :

Head -> 4 -> 5 -> 6 -> NULL

---- INSERTION OPTIONS ----

1. Insertion at the beginning

2. Insertion at the End

3. Insertion before a node

4. Insertion after a node

5. Insertion at a given position

Enter your choice: 1

Enter your data: 3

successfully entered at the beginning:

The Linked List is :

Head -> 3 -> 4 -> 5 -> 6 -> NULL

Press Enter to continue...

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 4

The Linked List is :

Head -> 3 -> 4 -> 5 -> 6 -> NULL

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 1

How many nodes you want to create : 3

Enter element 1 : 4

Enter element 2 : 5

Enter element 3 : 6

The Linked List is :

Head -> 4 -> 5 -> 6 -> NULL

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 2

The Linked List is :

Head -> 4 -> 5 -> 6 -> NULL

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 3

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 4

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

---- INSERTION OPTIONS ----

1. Insertion at the beginning

2. Insertion at the End

3. Insertion before a node

4. Insertion after a node

5. Insertion at a given position

Enter your choice: 4

Enter your data: 8

enter the position: 6

successfully entered after a node:

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL

Press Enter to continue...

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 4

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL

---- INSERTION OPTIONS ----

1. Insertion at the beginning

2. Insertion at the End

---- INSERTION OPTIONS ----

1. Insertion at the beginning

2. Insertion at the End

3. Insertion before a node

4. Insertion after a node

5. Insertion at a given position

Enter your choice: 2

Enter your data: 7

successfully entered at the end:

The Linked List is :

Head -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

Press Enter to continue...

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 4

The Linked List is :

Head -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

---- INSERTION OPTIONS ----

1. Insertion at the beginning

2. Insertion at the End

3. Insertion before a node

4. Insertion after a node

5. Insertion at a given position

Enter your choice: 3

Enter your data: 1

enter the position: 1

successfully entered before a node:

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

Press Enter to continue…

5 found at position 4

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 7

Before Sorting

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL

After Sorting

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 8

Linked List reversed successfully.

The Linked List is :

Head -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 1 -> NULL

Press Enter to continue...

3. Insertion before a node

4. Insertion after a node

5. Insertion at a given position

Enter your choice: 5

Enter your data: 9

enter the position: 8

successfully entered at position 8:

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> NULL

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 5

Enter the element you want to delete: 9

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> NULL 9 deleted from the Linked List

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL

Press Enter to continue…

----- SINGLE LINKED LIST -------

0. Exit

1. Create

2. Display

3. Count

4. Insertion

5. Deletion

6. Search

7. Sort

8. Reverse

Enter your choice: 6

Enter the item you want to search: 5

The Linked List is :

Head -> 1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL