**Search an Element**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void display (struct Node \*node)

{

while (node != NULL)

{

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

node = node->next;

}

printf ("\n");

}

int searchElement (struct Node \*head, int item)

{

struct Node \*current = head;

int index = 0;

while (current != NULL)

{

if (current->data == item)

{

return index;

}

current = current->next;

index++;

}

return -1;

}

int main ()

{

int item;

struct Node \*head = NULL;

struct Node \*node2 = NULL;

struct Node \*node3 = NULL;

struct Node \*node4 = NULL;

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

node2 = (struct Node \*) malloc (sizeof (struct Node));

node3 = (struct Node \*) malloc (sizeof (struct Node));

node4 = (struct Node \*) malloc (sizeof (struct Node));

head->data = 10;

head->next = node2;

node2->data = 15;

node2->next = node3;

node3->data = 20;

node3->next = node4;

node4->data = 25;

node4->next = NULL;

printf ("Linked List: ");

display (head);

printf ("Enter element to search: ");

scanf ("%d", &item);

int index = searchElement (head, item);

if (index == -1)

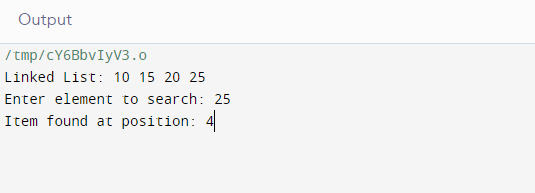
printf ("Item not found");

else

printf ("Item found at position: %d", index + 1);

return 0;

}



**Adding an Element at the beginning of a Linked list**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void insertBegin(struct Node\*\* head, int data)

{

struct Node\* new\_node=(struct Node\*)malloc(sizeof(struct Node));

new\_node->data=data;

new\_node->next=\*head;

\*head = new\_node;

printf("Inserted Item: %d\n", new\_node->data);

}

void printList(struct Node\* node)

{

printf("\nLinked List: ");

while(node != NULL)

{

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

node = node->next;

}

printf("\n");

}

int main()

{

struct Node\* head = NULL;

insertBegin(&head, 10);

insertBegin(&head, 15);

insertBegin(&head, 20);

insertBegin(&head, 25);

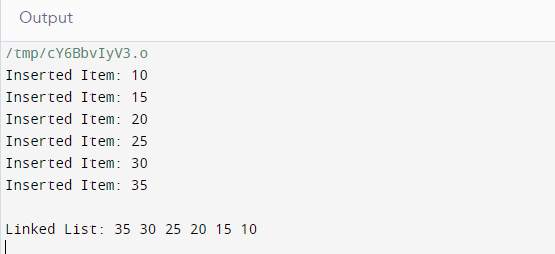
insertBegin(&head, 30);

insertBegin(&head, 35);

printList(head);

return 0;

}



**Inserting an Element in between a Linked list**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

int size = 0;

void insertMiddle (struct Node \*\*head, int data)

{

struct Node \*newNode = (struct Node \*) malloc (sizeof (struct Node));

newNode->data = data;

if (\*head == NULL)

{

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

size++;

return;

}

struct Node \*temp = \*head;

int mid = (size % 2 == 0) ? (size / 2) : (size + 1) / 2;

while (--mid)

{

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

size++;

}

void display (struct Node \*node)

{

printf ("Linked List : ");

while (node != NULL)

{

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

node = node->next;

}

printf ("\n\n");

}

int main ()

{

struct Node \*head = NULL;

struct Node \*node2 = NULL;

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

node2 = (struct Node \*) malloc (sizeof (struct Node));

head->data = 10;

head->next = node2;

node2->data = 50;

node2->next = NULL;

size = 2;

display (head);

insertMiddle (&head, 20);

display (head);

insertMiddle (&head, 40);

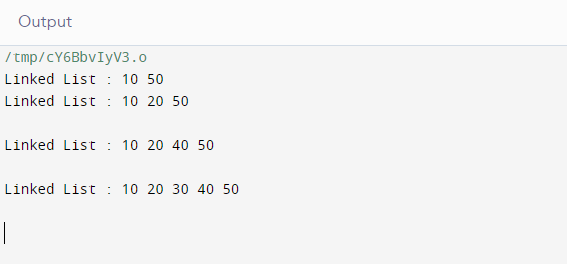
display (head);

insertMiddle (&head, 30);

display (head);

return 0;

}



**Inserting an Element at the end of a Linked list**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void insertEnd (struct Node \*\*head, int data)

{

struct Node \*newNode = (struct Node \*) malloc (sizeof (struct Node));

newNode->data = data;

newNode->next = NULL;

if (\*head == NULL)

{

\*head = newNode;

return;

}

struct Node \*temp = \*head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

printf ("%d inserted at the end\n", data);

}

void display (struct Node \*node)

{

while (node != NULL)

{

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

node = node->next;

}

printf ("\n");

}

int

main ()

{

struct Node \*head = NULL;

struct Node \*node2 = NULL;

struct Node \*node3 = NULL;

struct Node \*node4 = NULL;

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

node2 = (struct Node \*) malloc (sizeof (struct Node));

node3 = (struct Node \*) malloc (sizeof (struct Node));

node4 = (struct Node \*) malloc (sizeof (struct Node));

head->data = 22;

head->next = node2;

node2->data = 30;

node2->next = node3;

node3->data = 24;

node3->next = node4;

node4->data = 20;

node4->next = NULL;

printf ("Linked List Before Operations : ");

display (head);

insertEnd (&head, 5);

insertEnd (&head, 6);

insertEnd (&head, 7);

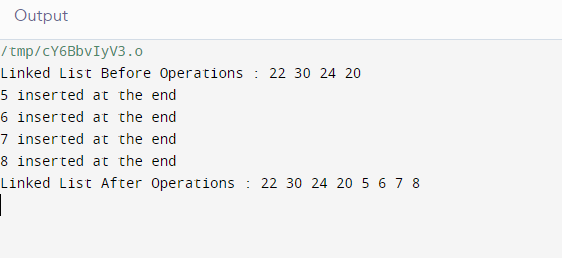
insertEnd (&head, 8);

printf ("Linked List After Operations : ");

display (head);

return 0;

}



**Insert an Element into a sorted Linked list**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*next;

};

void display (struct Node \*node)

{

while (node != NULL)

{

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

node = node->next;

}

}

struct Node \*newNode (int data)

{

struct Node \*newNode = (struct Node \*) malloc (sizeof (struct Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertion\_sort (struct Node \*\*head, struct Node \*newNode)

{

if (\*head == NULL || (\*head)->data >= newNode->data)

{

newNode->next = \*head;

\*head = newNode;

return;

}

struct Node \*current = \*head;

while (current->next != NULL && current->next->data < newNode->data)

current = current->next;

newNode->next = current->next;

current->next = newNode;

}

int main ()

{

int k;

struct Node \*head = NULL;

struct Node \*node2 = NULL;

struct Node \*node3 = NULL;

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

node2 = (struct Node \*) malloc (sizeof (struct Node));

node3 = (struct Node \*) malloc (sizeof (struct Node));

head->data = 10;

head->next = node2;

node2->data = 15;

node2->next = node3;

node3->data = 20;

node3->next = NULL;

printf ("Linked list before insertion : ");

display (head);

printf ("\nEnter data you want to insert: ");

scanf ("%d", &k);

insertion\_sort (&head, newNode (k));

printf ("Linked list after insertion : ");

display (head);

return 0;

}

