

Data Structure

Course code: CSE134

Instructor

Amatul Bushra Akhi

Assistant Professor

Department of CSE

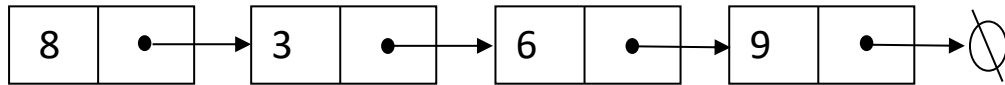
Topic: Deletion, Search, Counting

Team Member

<u>Name</u>	<u>ID</u>
Mehedi Hasan Saim	221-15-4844
Ishrat Zahan Easha	221-15-5212
Sohanur Rahman Saikat	221-15-4845
Habibur Rahman Habib	221-15-5174
Nibir Hossain	221-15-5057

Spring-2021

a) Consider the following linklist:



The "Node" contains integer data member "info" and a pointer member "link".
Write function for the following operation:

- 1) Find a data item in the list, and then insert a newnode next to it.
- 2) Delete a node from the nth position of the list.

Solution of (1):

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *link;
```

```
};
```

```
typedef struct node node;
```

```
node *head;
```

```
void display();
```

```
void search_data(int key);
```

```
void insert_end();
```

```
int main()
```

```
{
```

```
    node *node2, *node3, *node4;
```

```
    head = (node*)malloc(sizeof(node));
```

```
    node2 = (node*)malloc(sizeof(node));
```

```
    node3 = (node*)malloc(sizeof(node));
```

```
    node4 = (node*)malloc(sizeof(node));
```

```
    head->info = 8;
```

```
    head->link = node2;
```

```
    node2->info = 3;
```

```
    node2->link = node3;
```

```
    node3->info = 6;
```

```
    node3->link = node4;
```

```
    node4->info = 9;
```

```
    node4->link = NULL;
```

```
    display();
```

```
    int key;
```

```

printf("\nEnter search item: ");

scanf("%d", &key);

search_data(key);


insert_end();

display();

return 0;
}

void insert_end()
{
    int pos, i;

    node *temp, *newNode, *prev, *current;

    newNode = (node*) malloc (sizeof(node));

    temp = head;

    printf("\nEnter the position where you want to implement node: \n");

    scanf("%d",&pos);

    printf("\nEnter data for searching node: ");

    scanf("%d",&newNode->info);

    for(i=1; i<pos; i++)
    {
        prev = temp;

        current = temp->link;

```

```
        temp = temp->link;
    }
    newNode->link = current;
    prev->link = newNode;
}

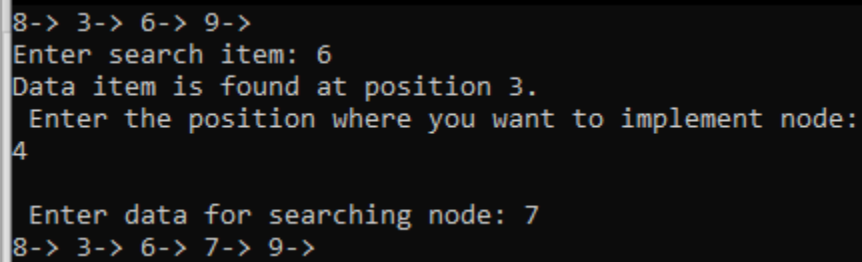
void display()
{
    node *temp;
    temp = head;
    while(temp!= NULL)
    {
        printf("%d-> ", temp->info);
        temp = temp->link;
    }
}

void search_data(int key)
{
    int pos=1;
    node *temp;
    temp = head;
    while(temp != NULL)
    {
```

```

    if(key == temp->info)
    {
        printf("Data item is found at position %d.", pos);
        break;
    }
    temp = temp->link;
    pos++;
}
}

```



```

8-> 3-> 6-> 9->
Enter search item: 6
Data item is found at position 3.
Enter the position where you want to implement node:
4
Enter data for searching node: 7
8-> 3-> 6-> 7-> 9->

```

Solution of (2):

```

#include<stdio.h>

struct node
{
    int info;
    struct node *link;
};

```

```
typedef struct node node;
```

```
node *head;
```

```
void display();
```

```
int main()
```

```
{
```

```
    node *node2, *node3, *node4;
```

```
    head = (node*)malloc(sizeof(node));
```

```
    node2 = (node*)malloc(sizeof(node));
```

```
    node3 = (node*)malloc(sizeof(node));
```

```
    node4 = (node*)malloc(sizeof(node));
```

```
    head->info = 8;
```

```
    head->link = node2;
```

```
    node2->info = 3;
```

```
    node2->link = node3;
```

```
    node3->info = 6;
```

```
    node3->link = node4;
```

```
    node4->info = 9;
```

```
    node4->link = NULL;
```

```
    display();
```

```

int pos;

node *temp, *prev, *current;

printf("\nEnter position you want to delete: ");

scanf("%d", &pos);

temp = head;

for(int i=1; i<pos; i++){

    prev = temp;

    current = temp-> link;

    temp = temp-> link;

}

prev-> link = current-> link;

free(current);

display();

return 0;

}

void display()

{

    node *temp;

    temp = head;

    while(temp!= NULL)

    {

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

```



```

    temp = temp->link;

}

}

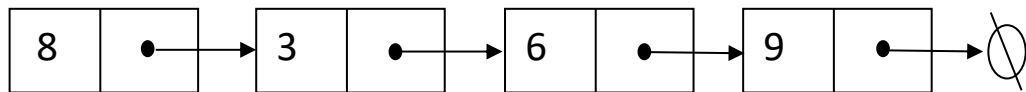
```

```

8-> 3-> 6-> 9->
Enter position you want to delete: 3
8-> 3-> 9->

```

Spring-2022



The “Element” contains integer data member “information” and a pointer member “link”. Write the function for the following operation—

1. Find a data item in the list.

Solution:

```

#include<stdio.h>

struct node
{
    int information;
    struct node * link;
};

typedef struct node node;

node *head;

void display();

```

int main()

{

node *node2, *node3, *node4;

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

node2 = (node*)malloc(sizeof(node));

node3 = (node*)malloc(sizeof(node));

node4 = (node*)malloc(sizeof(node));

head->information = 8;

head->link = node2;

node2->information = 3;

node2->link = node3;

node3->information = 6;

node3->link = node4;

node4->information = 9;

node4->link = NULL;

display();

int pos;

node *temp, *prev, *current;

printf("\nEnter position you want to delete: ");

scanf("%d", &pos);

```
temp = head;
for(int i=1; i<pos; i++){
    prev = temp;
    current = temp-> link;
    temp = temp-> link;
}
prev-> link = current-> link;
free(current);
display();
return 0;
}
```

void display()

```
{
    node *temp;
    temp = head;
    while(temp!= NULL)
    {
        printf("%d-> ", temp-> information);
        temp = temp->link;
    }
}
```

```
8-> 3-> 6-> 9->  
Enter position you want to delete: 3  
8-> 3-> 9->
```

Fall-2020

//Deletion and counting.....

3(b):

To prove your skills, create a Linked List with n Nodes and insert 10 in the list also. DataSoft Ltd. Doesn't like odd numbers. So, delete the odd numbers and show them how many nodes in the link exists. (*n= length of your student id*)

Solution:

```
#include<stdio.h>  
  
struct node  
{  
    int data;  
    struct node *next;  
};  
  
typedef struct node node;  
  
node *head;  
  
void createNode(int n);  
  
void display();  
  
void delete_odd();  
  
void insert();
```

int main()

```
{  
    int n;  
    printf("Enter Your id length for link list size: ");  
    scanf("%d", &n);  
  
    createNode(n);  
    insert();  
    display();  
    delete_odd();  
    printf("\nAfter deleting odd number");  
    display();  
  
    int nodeCount = countNodes();  
    printf("\nNumber of nodes in the linked list: %d\n", nodeCount);  
    return 0;  
}
```

void delete_odd()

```
{  
    node *temp, *prev, *current;  
    temp = head;  
    prev = temp;
```

```
current = temp->next;
while (current != NULL)
{
    if (current->data % 2 != 0)
    {
        prev->next = current->next;
        temp = current;
        current = current->next;
        free(temp);
    }
    else
    {
        prev = current;
        current = current->next;
    }
}

int countNodes()
{
    int count = 0;
    node *temp;
    temp = head;
```

```
while (temp != NULL)
{
    count++;
    temp = temp->next;
}
return count;
}

void insert()
{
    node *newNode;
    newNode = (node*) malloc(sizeof(node));
    newNode->data = 10;
    newNode->next = head;
    head = newNode;
}

void createNode(int n)
{
    head = (node*) malloc (sizeof(node));
    printf("Enter data for node 1: ");
    scanf("%d", &head->data);
    head->next = NULL;
```

```
node *temp, *newNode;

temp = head;

for(int i=2; i<=n; i++)
{
    newNode = (node*) malloc (sizeof(node));

    printf("Enter data for node %d: ",i);

    scanf("%d", &newNode->data);

    newNode->next=NULL;

    temp->next = newNode;

    temp = temp->next;
}

}

void display()
{
    printf("\n");

    node *temp;

    temp = head;

    while(temp!= NULL)
    {
        printf("%d-> ", temp->data);

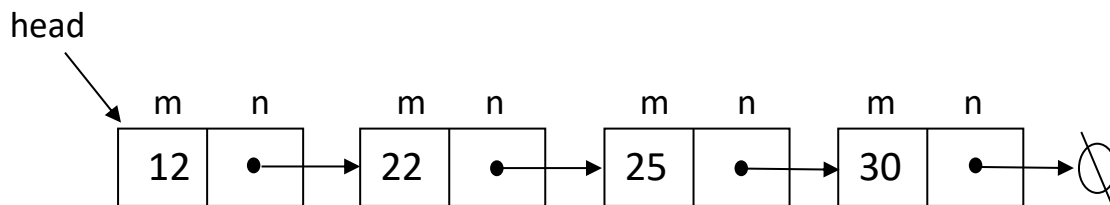
        temp = temp->next;
    }
}
```


}

```
Enter Your id length for link list size: 9
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 3
Enter data for node 4: 4
Enter data for node 5: 5
Enter data for node 6: 6
Enter data for node 7: 7
Enter data for node 8: 8
Enter data for node 9: 9

10-> 1-> 2-> 3-> 4-> 5-> 6-> 7-> 8-> 9->
After deleting odd number
10-> 2-> 4-> 6-> 8->
Number of nodes in the linked list: 5
```

Fall-2022



2(a). Construct a function to find a data item in the list and insert a new node after it.

Solution:

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

```
struct node
```

```
{
```

```
    int m;
```

```
    struct node *n;
```

```
};
```

```
typedef struct node node;
```

```
node *head;

void display();

void search_data(int key);

void insert_end();

int main()
{
    node *node2, *node3, *node4;

    head = (node*)malloc(sizeof(node));
    node2 = (node*)malloc(sizeof(node));
    node3 = (node*)malloc(sizeof(node));
    node4 = (node*)malloc(sizeof(node));

    head->m = 12;
    head->n = node2;
    node2->m = 22;
    node2->n = node3;
    node3->m = 25;
    node3->n = node4;
    node4->m = 30;
    node4->n = NULL;

    display();
```

```
int key;

printf("\nEnter search item: ");

scanf("%d", &key);

search_data(key);


insert_end();

display();

return 0;
}

void insert_end()
{
    node *temp, *newNode;

    newNode = (node*) malloc (sizeof(node));

    printf("Enter the data for last node: ");

    scanf("%d", &newNode->m);

    newNode->n = NULL;

    temp = head;

    while(temp->n != NULL)
    {
        temp = temp->n;
    }

    temp->n = newNode;
```

```
}
```

```
void display()
```

```
{
```

```
    node *temp;
```

```
    temp = head;
```

```
    while(temp!= NULL)
```

```
    {
```

```
        printf("%d-> ", temp->m);
```

```
        temp = temp->n;
```

```
    }
```

```
}
```

```
void search_data(int key)
```

```
{
```

```
    int pos=1;
```

```
    node *temp;
```

```
    temp = head;
```

```
    while(temp != NULL)
```

```
    {
```

```
        if(key == temp->m)
```

```
        {
```

```
            printf("\nData item is found at position %d.\n\n", pos);
```

```
            break;
```

```

    }

    temp = temp->n;

    pos++;

}

}

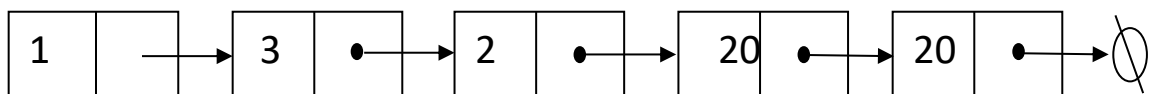
```

```

12-> 22-> 25-> 30->
Enter search item: 25
Data item is found at position 3.
Enter the data for last node: 35
12-> 22-> 25-> 30-> 35->

```

Summer-2020



2(c). Delete the last Node of the link list.

2(d). Count how many Nodes in the link list.

Solution-2(c):

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *next;
```

```
};
```

```
typedef struct node node;

node *head;

void display();

void delete_end();

int main()
{
    node *second, *third, *fourth, *fifth;
    head = (node*)malloc(sizeof(node));
    second = (node*)malloc(sizeof(node));
    third = (node*)malloc(sizeof(node));
    fourth= (node*)malloc(sizeof(node));
    fifth = (node*)malloc(sizeof(node));

    head->info = 1;
    head->next = second;
    second->info = 3;
    second->next = third;
    third->info = 2;
    third->next = fourth;
    fourth->info = 20;
    fourth->next = fifth;
    fifth->info = 20;
```

```
fifth->next = NULL;
```

```
display();
```

```
delete_end();
```

```
printf("\nAfter deleting last node: ");
```

```
display();
```

```
return 0;
```

```
}
```

```
void display()
```

```
{
```

```
    printf("\n");
```

```
    node *temp;
```

```
    temp = head;
```

```
    while(temp!= NULL)
```

```
    {
```

```
        printf("%d-> ", temp->info);
```

```
        temp = temp->next;
```

```
    }
```

```
}
```

```
void delete_end()
```

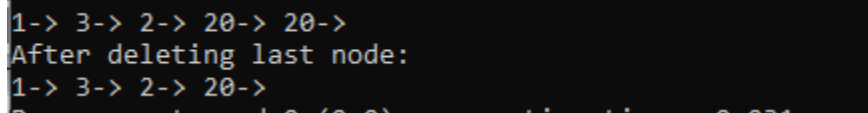
```
{
```

```
    node *temp, *prev;
```

```

temp = head;
while(temp->next != NULL)
{
    prev = temp;
    temp = temp->next;
}
prev->next = NULL;
free(temp);
}

```



1-> 3-> 2-> 20-> 20->
After deleting last node:
1-> 3-> 2-> 20->

Solution-2(d):

```

#include<stdio.h>

struct node
{
    int info;
    struct node *next;
};

typedef struct node node;
node *head;

```



```
void display();  
void count();  
int main()  
{  
    node *second, *third, *fourth, *fifth;  
    head = (node*)malloc(sizeof(node));  
    second = (node*)malloc(sizeof(node));  
    third = (node*)malloc(sizeof(node));  
    fourth = (node*)malloc(sizeof(node));  
    fifth = (node*)malloc(sizeof(node));  
  
    head->info = 1;  
    head->next = second;  
    second->info = 3;  
    second->next = third;  
    third->info = 2;  
    third->next = fourth;  
    fourth->info = 20;  
    fourth->next = fifth;  
    fifth->info = 20;  
    fifth->next = NULL;
```

```
    display();  
    count();  
    return 0;  
}  
  
void count()  
{  
    int count =0;  
    node *temp;  
    temp = head;  
    while(temp != NULL)  
    {  
        temp = temp->next;  
        count++;  
    }  
    printf("\nTotal Number of node is %d\n", count);  
}  
  
void display()  
{  
    printf("\n");  
    node *temp;  
    temp = head;  
    while(temp!= NULL)
```

```

{
    printf("%d-> ", temp->info);
    temp = temp->next;
}
}

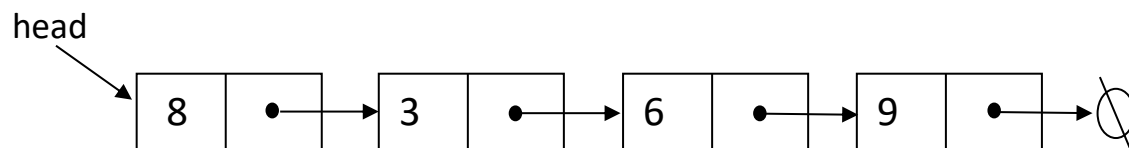
```

```

1-> 3-> 2-> 20-> 20->
Total Number of node is 5

```

Summer-2021



(a). The “Node” contains integer data member “your first name” and a pointer member “your last name”. Write the function to find a data item in the list, and then insert a new node before of it.

Solution:

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

```
struct node
```

```
{
```

```
    int mehedi;
```

```
    struct node *saim;
```

```
};
```

```
typedef struct node node;
```

```
node *head;

void display();

void search_data(int key);

int main()
{
    node *node2, *node3, *node4;

    head = (node*)malloc(sizeof(node));
    node2 = (node*)malloc(sizeof(node));
    node3 = (node*)malloc(sizeof(node));
    node4 = (node*)malloc(sizeof(node));

    head->mehedi = 8;
    head->saim = node2;
    node2->mehedi = 3;
    node2->saim = node3;
    node3->mehedi = 6;
    node3->saim = node4;
    node4->mehedi = 9;
    node4->saim = NULL;

    display();

    int key;
```

```
printf("\nEnter search item: ");  
  
scanf("%d", &key);  
  
search_data(key);  
  
insert_before();  
  
display();  
  
return 0;  
}  
  
void display()  
{  
    node *temp;  
    temp = head;  
    while(temp!= NULL)  
    {  
        printf("%d-> ", temp->mehedi);  
        temp = temp->saim;  
    }  
}  
  
void search_data(int key)  
{  
    int pos=1;  
    node *temp;
```

```

temp = head;
while(temp != NULL)
{
    if(key == temp->mehedi)
    {
        printf("Data item is found at position %d.", pos);
        break;
    }
    temp = temp->saim;
    pos++;
}
}

void insert_before()
{
    int pos, i;
    node *temp, *newNode, *prev, *current;
    newNode = (node*) malloc (sizeof(node));
    printf("\nEnter the position where you want to implement node: ");
    scanf("%d",&pos);

    printf("\nEnter data for newNode: ");
    scanf("%d",&newNode->mehedi);

```

```
temp = head;
for(i=0; i<pos; i++)
{
    prev = temp;
    current = temp->saim;
    temp = temp->saim;
}
newNode->saim = current;
prev->saim = newNode;
}
```

```
8-> 3-> 6-> 9->
Enter search item: 6
Data item is found at position 3.
Enter the position where you want to implement node: 2

Enter data for newNode: 5
8-> 3-> 5-> 6-> 9->
```