**EXPERIMENT-5**

**AIM:** Create a linked list with nodes having info about a student and perform-

1. Insert a new node at specified position.
2. Delete of a node with the roll number of student specified.
3. Reversal of that linked list

**ALGORITHM:**

**SOURCE CODE:**

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int roll;

string Name;

string Dept;

int Marks;

Node\* next;

};

Node\* head = new Node();

bool check(int x)

{

if (head == NULL)

return false;

Node\* t = new Node;

t = head;

while (t != NULL) {

if (t->roll == x)

return true;

t = t->next;

}

return false;

}

void Insert\_Record(int roll, string Name,string Dept, int Marks)

{

if (check(roll)) {

cout << "Student with this \n"<< "record Already Exists\n";

return;

}

Node\* t = new Node();

t->roll = roll;

t->Name = Name;

t->Dept = Dept;

t->Marks = Marks;

t->next = NULL;

if (head == NULL|| (head->roll >= t->roll)) {

t->next = head;

head = t;

}

else {

Node\* c = head;

while (c->next != NULL && c->next->roll < t->roll) {

c = c->next;

}

t->next = c->next;

c->next = t;

}

cout << "Record Inserted "<< "Successfully\n";

}

int Delete\_Record(int roll)

{

Node\* t = head;

Node\* p = NULL;

if (t != NULL&& t->roll == roll) {

head = t->next;

delete t;

cout << "Record Deleted "<< "Successfully\n";

return 0; }

while (t != NULL && t->roll != roll) {

p = t;

t = t->next; }

if (t == NULL) {

cout << "Record does not Exist\n";

return -1; }

else{

p->next = t->next;

delete t;

cout << "Record Deleted "

<< "Successfully\n";

return 0;}

}

void Show\_Record()

{

Node\* p = head;

if (p == NULL) {

cout << "No Record "<< "Available\n"; }

else {

cout << "Index\tName\tCourse"<< "\tMarks\n";

while (p != NULL) {

cout << p->roll << " \t" << p->Name << "\t" << p->Dept << "\t" << p->Marks << endl;

p = p->next; }

}

}

Node\* reverse()

{

Node\* current = head;

Node \*prev = NULL, \*next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next; }

head = prev;

return head;

}

int main()

{

head = NULL;

string Name, Course;

int Roll, Marks;

while (true) {

cout << "1 create a new Record\n"

"2 delete a student record\n"

"3 view all students record\n"

"4 revese the link list\n"

"5 Exit\n";

cout << "\nEnter your Choice\n";

int Choice;

cin >> Choice;

if (Choice == 1) {

cout << "Enter Name of Student\n";

cin >> Name;

cout << "Enter Roll Number of Student\n";

cin >> Roll;

cout << "Enter Course of Student \n";

cin >> Course;

cout << "Enter Total Marks of Student\n";

cin >> Marks;

Insert\_Record(Roll, Name, Course, Marks);

}

else if (Choice == 2) {

cout << "Enter Roll Number of Student whose record is to be deleted\n";

cin >> Roll;

Delete\_Record(Roll);

}

else if (Choice == 3) {

Show\_Record();

}

else if (Choice == 4) {

Node\*newhead=reverse();

Show\_Record();

}

else if (Choice == 5) {

exit(0);

}

else {

cout << "Invalid Choice " << "Try Again\n";

}

}

return 0;

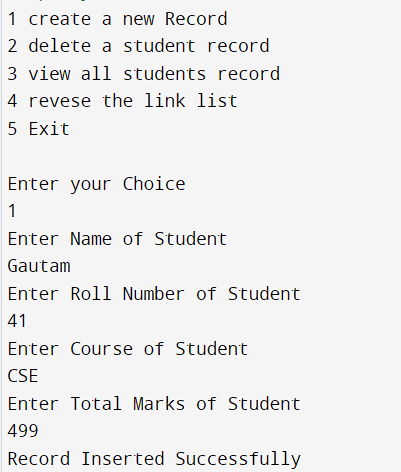
}

**EXPERIMENT-5**

**AIM:** Create a linked list with nodes having info about a student and perform-

1. Insert a new node at specified position.
2. Delete of a node with the roll number of student specified.
3. Reversal of that linked list

**OUTPUT:**

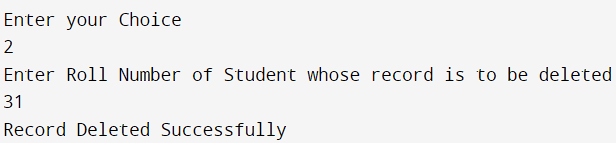
****

**Graphical user interface, text

Description automatically generated**

**A screenshot of a computer

Description automatically generated with low confidence**

****

**EXPERIMENT-6**

**AIM:** Create doubly linked list with nodes having information about an employee and perform Insertion at front of doubly linked list and perform deletion at end of that doubly linked list.

**ALGORITHM:**

**SOURCE CODE:**

#include<iostream>

using namespace std;

class node{

public:

string name;

int phoneno;

int ID;

node\* prev;

node\* next;

};

node\* head=new node;

bool check(int x){

if (head == NULL)

return false;

node\* t = new node;

t = head;

while (t != NULL) {

if (t->ID == x)

return true;

t = t->next;}

return false;

}

void push(string name,int id,int phone){

if (check(id)) {

cout << "Employee with this record Already Exists\n";

return;}

node\* new\_rec=new node;

new\_rec->name=name;

new\_rec->phoneno=phone;

new\_rec->ID=id;

new\_rec->next=head;

new\_rec->prev=NULL;

if (head!=NULL){

head->prev=new\_rec;

new\_rec->next=head;}

head=new\_rec;

}

void pop(int id){

node\* del=new node;

del=head;

while(del!=NULL){

if (del->ID== id)

break;

del = del->next;}

if (head == NULL || del == NULL)

return;

if (head == del){

head = del->next;}

if (del->next != NULL)

del->next->prev = del->prev;

if (del->prev != NULL)

del->prev->next = del->next;

free(del);

return;

}

int main(){

head=NULL;

string name;

int id,phone;

while(true){

cout<<"1 Insert a record"<<endl;

cout<<"2 Delete a record"<<endl;

cout<<"3 Exit"<<endl;

int ch;

cout<<"Enter your choice:"<<endl;

cin>>ch;

if (ch==1){

cout<<"Name of Employee:"<<endl;

cin>>name;

cout<<"Phone number:"<<endl;

cin>>phone;

cout<<"Employee id:"<<endl;

cin>>id;

}

else if (ch==2){

cout<<"Enter id to delete:"<<endl;

int n;

cin>>n;

pop(n);}

else if (ch == 3) {

break;}

else {

cout << "Invalid Choice " << "Try Again\n";

break;}

}

return 0;

}

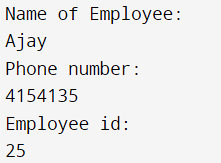
**EXPERIMENT-6**

**AIM:** Create doubly linked list with nodes having information about an employee and perform Insertion at front of doubly linked list and perform deletion at end of that doubly linked list.

**OUTPUT:**

**Text

Description automatically generated**

****

**Text

Description automatically generated with medium confidence**