**TASK 1:**

#include<iostream>

using namespace std;

struct node{

int data;

node\* next;

};

class parent{

node\* head = nullptr;

node\* curr = nullptr;

public:

parent(){

head = nullptr;

curr = nullptr;

}

bool isEmpty(){

return head == NULL;

}

void create\_node(int value){

node\* temp = new node;

temp->data = value;

temp->next = NULL;

if (head == NULL){

head = curr = temp;

}

else{

curr->next = temp;

curr = curr->next;

}

}

void beginning(){

int x;

cout << "Enter the value you want to add at start: ";

cin >> x;

node\* temp = new node;

temp->data = x;

temp->next = head;

head = temp;

}

void xposition(){

int x;

cout << "Enter the value you want to add at middle: ";

cin >> x;

node\* temp1 = head;

node\* temp = new node;

node\* temp2 = temp1;

cout << "Enter the Position you want to insert the Data ";

cout << endl;

int pos;

cin >> pos;

int count = 0;

if (pos == 0) {

temp->data = x;

temp->next = temp1;

head = temp;

}

else {

while (temp1 != NULL) {

if (count == pos) {

temp1 = temp1->next;

temp->data = x;

temp2->next = temp;

temp->next = temp1;

break;

}

else {

temp2 = temp;

temp1 = temp1->next;

count = count + 1;

}

}

}

}

void last(){

int x;

cout << "Enter the value you want to add at end: ";

cin >> x;

node\* newNode = new node();

newNode->data = x;

newNode->next = NULL;

if (head == NULL){

head = newNode;

}

else{

node\* temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

}

}

void print(){

cout << "Linked list is: " << endl;

node\* temp = head;

while (temp != NULL){

cout << temp->data << endl;

temp = temp->next;

}

}

};

int main(){

parent var;

int choice = 0;

var.create\_node(5);

var.create\_node(3);

var.create\_node(1);

var.create\_node(2);

var.create\_node(6);

while (choice != 5){

cout << "1- put the node at beginning " << endl;

cout << "2- put node at last " << endl;

cout << "3- put node at position x " << endl;

cout << "4- Display linked list " << endl;

cout << "Enter your choice : ";

cin >> choice;

switch (choice) {

case 1:

var.beginning();

break;

case 2:

var.last();

break;

case 3:

var.xposition();

break;

case 4:

var.print();

break;

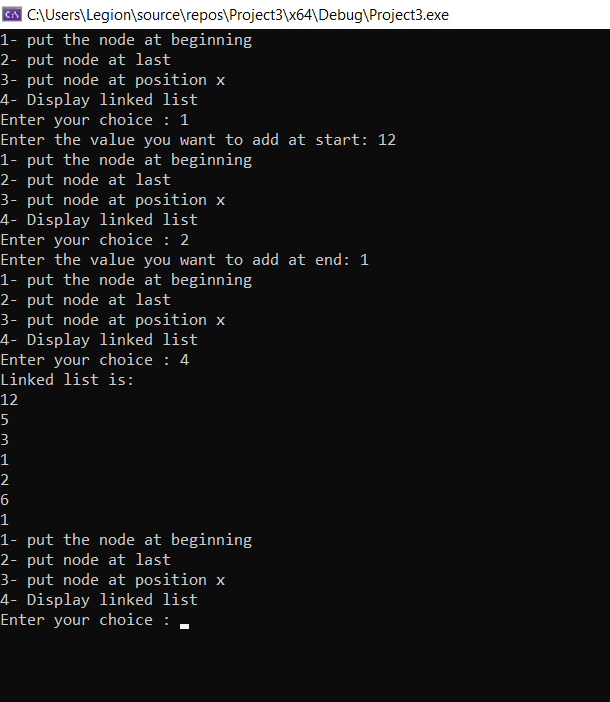
default:

break;

}

}

}



**TASK 2:**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

void push(Node\*\* head\_ref, int new\_data){

Node\* new\_node = new Node();

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void deleteNode(Node\*\* head\_ref, int key){

Node\* temp = \*head\_ref;

Node\* prev = NULL;

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

\*head\_ref = temp->next;

delete temp;

return;

}

else{

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

prev = temp;

temp = temp->next;

}

if (temp == NULL)

return;

prev->next = temp->next;

delete temp;

}

}

void printList(Node\* node){

while (node != NULL)

{

cout << node->data << " ";

node = node->next;

}

}

int main(){

Node\* head = NULL;

push(&head, 7);

push(&head, 1);

push(&head, 3);

push(&head, 2);

puts("Created Linked List: ");

printList(head);

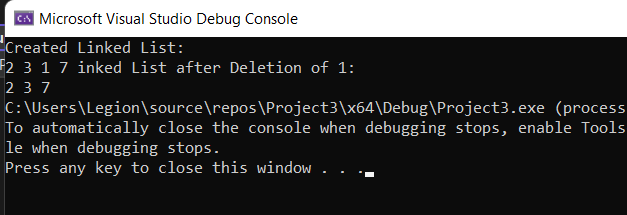
deleteNode(&head, 1);

puts("inked List after Deletion of 1: ");

printList(head);

return 0;

}



TASK 3:

#include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* next;

Node(int data)

{

this->data = data;

next = NULL;

}

};

struct LinkedList {

Node\* head;

LinkedList() { head = NULL; }

void reverse()

{

Node\* current = head;

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

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

void print()

{

struct Node\* temp = head;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

}

void push(int data)

{

Node\* temp = new Node(data);

temp->next = head;

head = temp;

}

};

int main()

{

LinkedList VAR;

VAR.push(20);

VAR.push(4);

VAR.push(15);

VAR.push(85);

cout << "Given linked list\n";

VAR.print();

VAR.reverse();

cout << "\nReversed Linked list \n";

VAR.print();

return 0;

}

