TASK 1:

#include <iostream>

#include <string>

using namespace std;

class Array {

public:

int item;

Array\* next;

Array() { //default constructor

this->next = 0; //initalizing with 0

}

};

class LinkList {

public:

Array\* head; //array class object

LinkList() { //default constructor of class linklist

this->head = 0; //initalizing head with 0

}

LinkList(Array\* head) { //parametrized constructor

this->head = head; //using this pointer to assign value at this instant

}

LinkList(LinkList& linkList) {

this->head = linkList.head;

}

~LinkList() { //destructor

delete head; //deleting the dynamic memory that we allocated

}

void PrintList() { //function to print the array/list

cout << "Link List Items: "<<endl;

Array\* tempArray = this->head;

while (tempArray != 0) { //while loop until temparray is not equal to 0

cout << tempArray->item << " ";

tempArray = tempArray->next;

}

}

bool Insert\_Element(int X) { //function to insert elements in array

Array\* newArray = new Array();

newArray->item = X;

if (this->head == 0) {

this->head = newArray;

}

else {

Array\* tempArray = this->head;

while (tempArray->next != 0)

{

tempArray = tempArray->next;

}

tempArray->next = newArray;

}

return true;

}

int Length() { //function to check length

Array\* tempArray = head;

int numberArrays = 0;

while (tempArray != 0) {

numberArrays++; //increment in the length of list

tempArray = tempArray->next;

}

return numberArrays; //returning the length of the list

}

Array\* getArrayByPosition(int position) {

if (position >= 0) {

Array\* currentArray = head;

int i = 0;

for (; i < position; ++i) {

currentArray = currentArray->next;

}

return currentArray;

}

return 0;

}

int Insert\_Element\_at(int X, int pos) { //function to insert element at a certain index

Array\* newArray = new Array();

newArray->item = X;

newArray->next = 0;

if (pos == 0) {

newArray->next = head;

head = newArray;

}

else if (pos == Length()) {

Insert\_Element(X);

}

else {

Array\* prevArray = getArrayByPosition(pos - 1);

Array\* nextArray = getArrayByPosition(pos);

prevArray->next = newArray;

newArray->next = nextArray;

return 0;

}

}

int Delete\_Element(int X) { //function to delete an element in an array

if (this->head->item == X) {

head = head->next;

}

else {

Array\* tempArray = head;

Array\* tempPrevious = head;

bool isFound = false;

while (!(isFound = tempArray->item == X) && tempArray->next != 0) {

tempPrevious = tempArray;

tempArray = tempArray->next;

}

if (isFound) {

tempPrevious->next = tempArray->next; //if the number is found in list then delete it

}

}

return 0;

}

bool is\_Empty() {

return (this->head == 0); //if list is empty it will display empty

}

};

int main() {

LinkList\* linkList = new LinkList();

int ch = -1;

int number;

int position;

while (ch != 6) { //below is the main driver program to give user defined operations

cout << " 1. Call Insert Element" << endl;

cout << " 2. Call Insert Element at position" << endl;

cout << " 3. Call Print List" << endl;

cout << " 4. Call Delete Element" << endl;

cout << " 5. Call Check if List is Empty" << endl;

cout << "6. Call Exit" << endl;

cout << "Your choice: ";

cin >> ch;

cout << "\n";

switch (ch) { //using switch statement

//below are just cases. case1, case2, case3 and so on till case 6.

//each case performs and calls functions accoring to the menu given above

case 1:

cout << "Enter the number to insert : ";

cin >> number;

linkList->Insert\_Element(number);

break;

case 2:

cout << "Enter the number to insert : ";

cin >> number;

cout << "Enter the position on which to insert : ";

cin >> position;

linkList->Insert\_Element\_at(number, position);

break;

case 3:

linkList->PrintList();

break;

case 4:

{

cout << "Enter the number to delete : ";

cin >> number;

linkList->Delete\_Element(number);

cout << "The number '" << number << "' is now deleted " << endl;

break;

}

break;

case 5:

if (linkList->is\_Empty()) {

cout << "The list is empty";

}

else {

cout << "The list isn't empty";

}

break;

case 6:

delete linkList;

break;

default:

break;

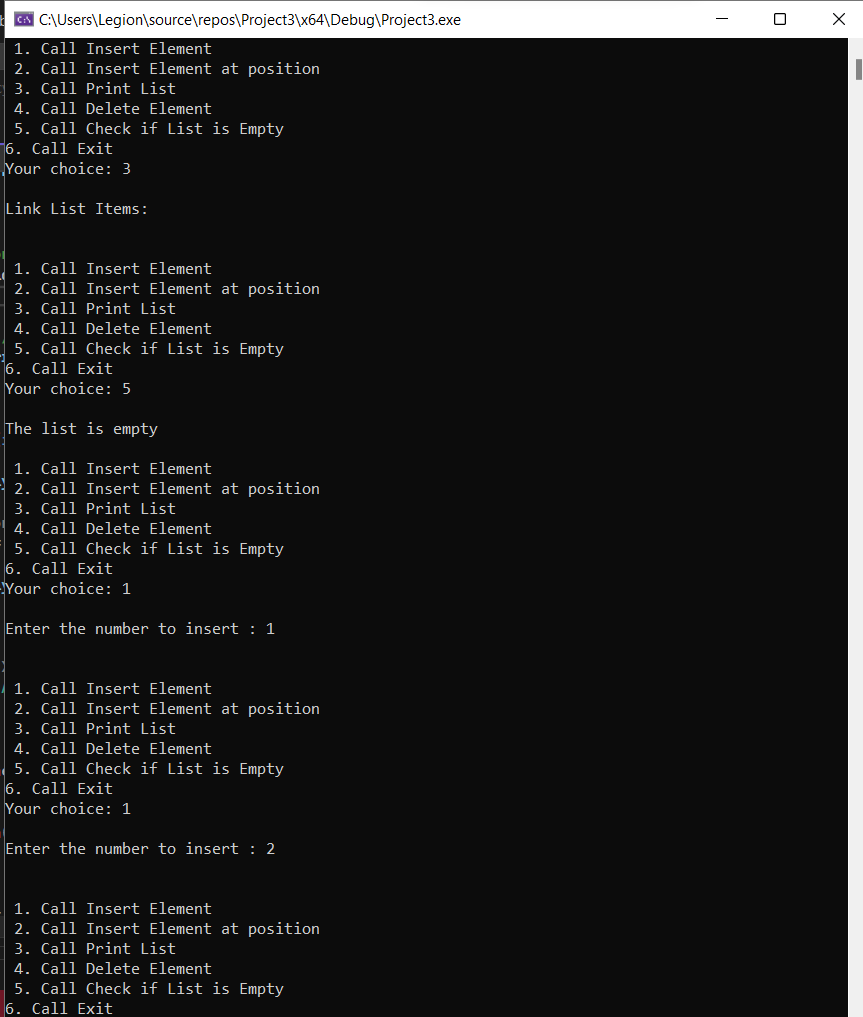
}

cout << "\n\n";

}

return 0;

}



Text

Description automatically generated

Text

Description automatically generated

TASK 2:

#include<iostream>

using namespace std;

class linkedlist {

public:

int data;

linkedlist\* next;

linkedlist(int d) {

data = d;

next = 0;

}

};

void printll(const linkedlist\* head) {

while (head != 0) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

linkedlist\* elementsentry(int n) {

int x;

linkedlist\* head = 0;

linkedlist\* tail = 0;

cout << endl;

for (int i = 0; i < n; i++) {

cout << "enter element number " << i + 1 << ": ";

cin >> x;

linkedlist\* n = new linkedlist(x);

if (head == 0) {

head = n;

tail = n;

}

else {

tail->next = n;

tail = tail->next;

}

}

return head;

}

void duplicatechecker(linkedlist\* check) {

linkedlist\* ptr1, \* ptr2, \* dup;

ptr1 = check;

while (ptr1 != 0 && ptr1->next != 0) {

ptr2 = ptr1;

while (ptr2->next != 0) {

if (ptr1->data == ptr2->next->data) {

dup = ptr2->next;

ptr2->next = ptr2->next->next;

delete(dup);

}

else

ptr2 = ptr2->next;

}

ptr1 = ptr1->next;

}

}

int main() {

int n;

cout << "Enter how many elements you want to enter : ";

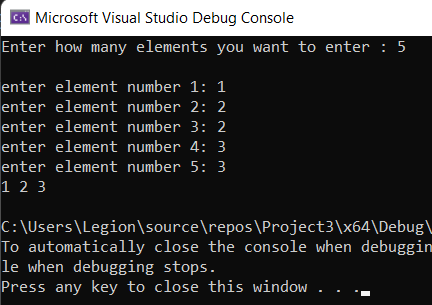
cin >> n;

linkedlist\* head = elementsentry(n);

duplicatechecker(head);

printll(head);

}



TASK 3:

#include<iostream>

#include<stdio.h>

using namespace std;

typedef struct nodee {

int data;

nodee\* next;

nodee\* prev;

}node;

node\* addnode(node\* f) { //to add nodes element at beginning

node\* n;

int ch;

n = new node;

cout << "enter the element to insert at beginning :";

cin >> n->data; //entering the element at beggining

n->next = n->prev = 0;

f = n;

return f;

}

void printnode(node\* f) { //to display the element

node\* temp = f;

if (temp == 0) { //if 0 then we know that list is empty

cout << endl << "empty list" <<endl;

return;

}

else { //else we will display the node

cout << "Node is ";

cout << temp->data; //displaying the node that we entered

temp = temp->next;

cout << endl;

}

}

node\* deelete(node\* f) {

node\* temp1, \* temp2; int ch;

temp1 = temp2 = f;

if (temp1 == 0) { // if temp1 is 0 then it means list is empty and nothing can be deleted from empty list

cout << "nothing can be deleted from empty list" << endl;

}

else {

f = f->next;

cout << "deleted"<<endl;

}

return f;

}

node\* head;

int main() {

int choice;

choice = 1;

head = 0;

while (choice != 0) { //menu will not exit unless user enters 0

cout << "1- Add node "<<endl;

cout << "2- Delete nodes "<<endl;

cout << "3- Display node "<<endl;

cout << "0- QUIT "<<endl;

cin >> choice;

switch (choice) {

case 1:

head = addnode(head);

break;

case 2:

head = deelete(head);

break;

case 3:

printnode(head);

break;

case 0:

choice = 0;

break;

default:

cout << "invalid"<<endl;

break;

}

}

return 0;

}

Text

Description automatically generated