TASK 1:

#include <iostream>

#include <string>

using namespace std;

class Array{

public:

int item;

Array\* next;

Array() { //default constructor

this->next = NULL; //initalizing with null

}

};

class LinkList{

public:

Array\* head; //array class object

LinkList(){ //default constructor of class linklist

this->head = NULL; //initalizing with null

}

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:\n";

Array\* tempArray = this->head;

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

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

tempArray = tempArray->next;

}

}

int search\_Element(int X) { //function to search for a certain element in array

Array\* tempArray = head;

int numberArrays = 0;

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

if (tempArray->item == X) { //if equal then return that number

return numberArrays;

}

numberArrays++;

tempArray = tempArray->next;

}

return -1;

}

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

Array\* newArray = new Array();

newArray->item = X;

if (this->head == NULL){

this->head = newArray;

}

else{

Array\* tempArray = this->head;

while (tempArray->next != NULL)

{

tempArray = tempArray->next;

}

tempArray->next = newArray;

}

return true;

}

int Length() { //function to check length

Array\* tempArray = head;

int numberArrays = 0;

while (tempArray != NULL){

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 NULL;

}

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 = NULL;

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 != NULL){

tempPrevious = tempArray;

tempArray = tempArray->next;

}

if (isFound){

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

}

else{

cout << "Item not found!\n"; //if not found then display this message

}

}

return 0;

}

bool is\_Empty() {

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

}

void Print\_Reverse\_List() {

Array\* current = head;

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

while (current != NULL) { //below is the logic to reverse the elements present in list. It works like swapping

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

PrintList(); //calling this printlist function to print the new reversed list

}

void Empty\_List() {

this->head = NULL;

}

};

int main() {

LinkList\* linkList = new LinkList();

int ch = -1;

int number;

int position;

while (ch != 10) { //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 Search Element" << endl;

cout << " 5. Call Delete Element" << endl;

cout << " 6. Call Check if List is Empty" << endl;

cout << " 7. Call Display Length of List" << endl;

cout << " 8. Call Print Reverse List" << endl;

cout << " 9. Call Clear List" << endl;

cout << "10. 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 10.

//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 search : ";

cin >> number;

int index = linkList->search\_Element(number);

if (index != -1) {

cout << "The element exists in position " << index;

}

else {

cout << "The element doesn't exist "<<endl;

}

}

break;

case 5:

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

cin >> number;

linkList->Delete\_Element(number);

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

break;

case 6:

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

cout << "The list is empty";

}

else {

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

}

break;

case 7:

cout << "Length of array is: " << linkList->Length();

break;

case 8:

linkList->Print\_Reverse\_List();

break;

case 9:

linkList->Empty\_List();

cout << "List is empty";

break;

case 10:

delete linkList;

break;

default:

break;

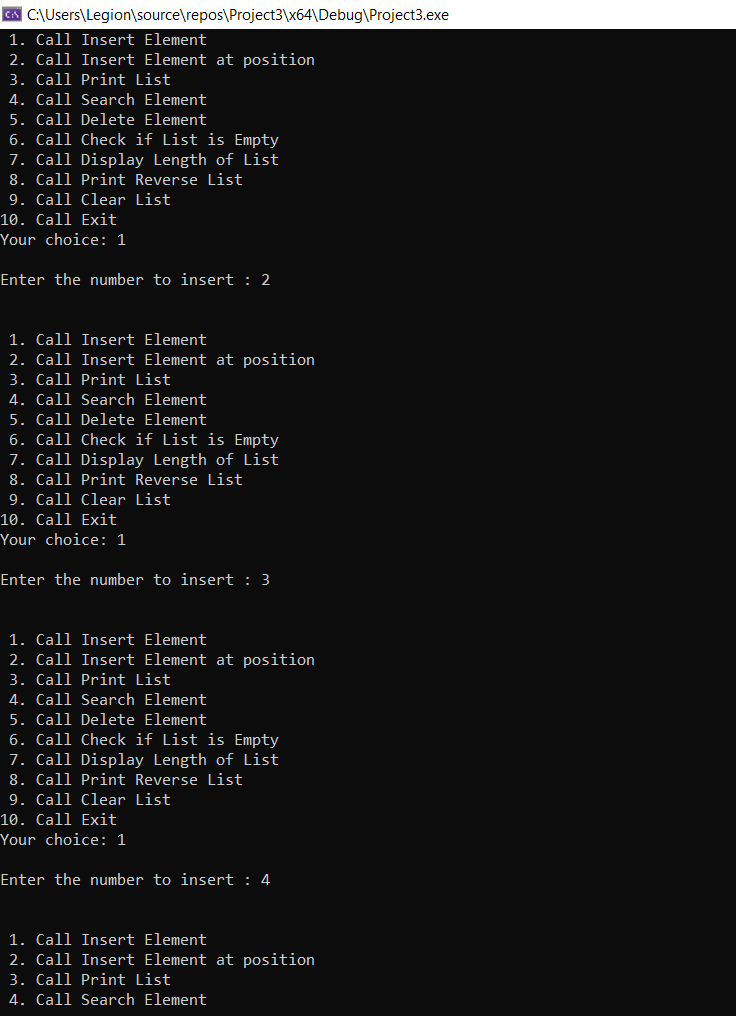
}

cout << "\n\n";

}

return 0;

}



Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated