TASK 1:

#include <iostream>

using namespace std;

int main(){

int size;

cout << "Enter the number elements you want to enter in array : ";

cin >> size;

int\* array = new int[size];

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

cin >> array[i];

}

cout << "Original Array is : ";

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

cout << array[i] << " ";

}

int oddInd = 1;

int evenInd = 0;

while (1){

while (evenInd < size && array[evenInd] % 2 == 0)

evenInd += 2;

while (oddInd < size && array[oddInd] % 2 == 1)

oddInd += 2;

if (evenInd < size && oddInd < size)

swap(array[evenInd], array[oddInd]);

else

break;

}

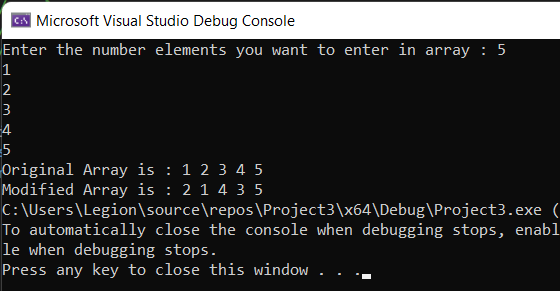
cout<< endl << "Modified Array is : ";

for (int i = 0; i < size; i++)

cout << array[i] << " ";

return 0;

}



TASK 2:

#include <iostream>

#include <limits>

#define MAX 200

using namespace std;

int main() {

int arr[MAX];

int n, i, j, k, temp;

cout << "Enter the total number of numbers to read: ";

cin >> n;

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

cout << "Enter element [" << i + 1 << "] ";

cin >> arr[i];

cout << endl;

}

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

cout << "Factors of " << arr[i] << " are: " << endl;

for (k = 1; k <= arr[i]; ++k){

if (arr[i] % k == 0)

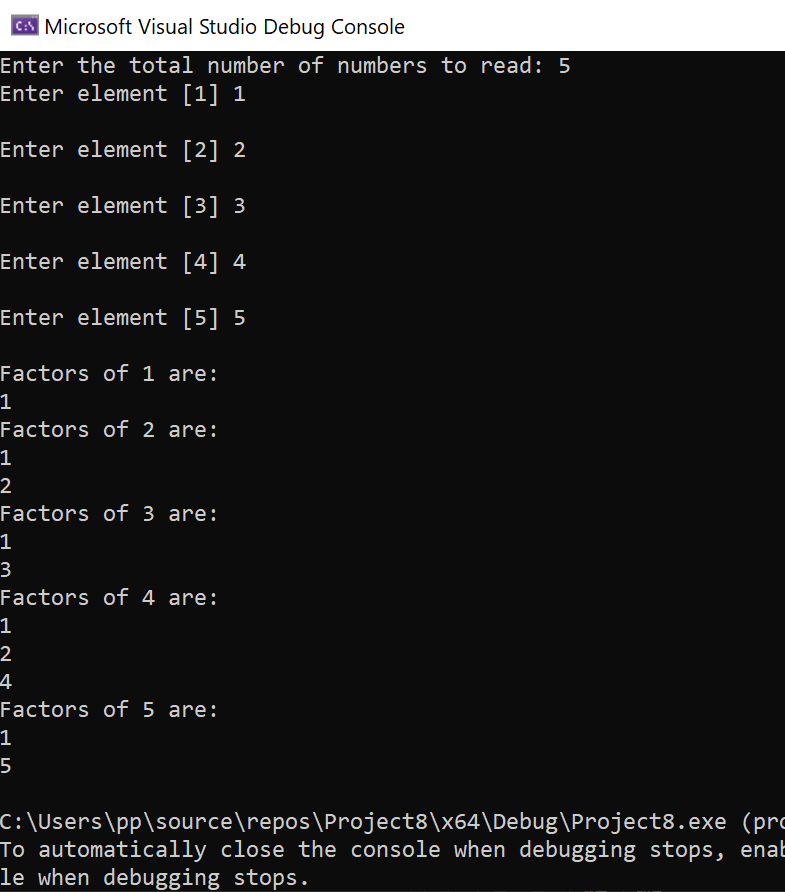
cout << k << endl;

}

}

return 0;

}



TASK 2:

#include <iostream>

using namespace std;

void printDivisors(int\* jk, int a){

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

for (int j = 1; j < jk[i]; j++){

if (jk[i] % j == 0){

if (j == 1){

cout << jk[i] << ": " << j << " ";

}

else{

cout << j << " ";

}

}

}cout << endl;

}

}

int main(){

int size;

cout << "Enter the number elements you want to enter in array : ";

cin >> size;

int\* array = new int[size];

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

cin >> array[i];

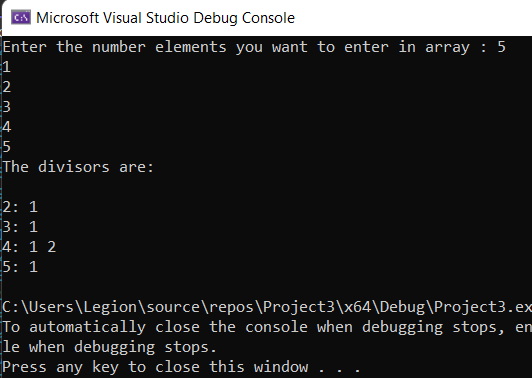
}

cout << "The divisors are: \n";

printDivisors(array, size);

return 0;

}



TASK 3:

#include<iostream>

using namespace std;

struct node {

int data;

node\* next;

};

node\* head = NULL;

node\* head1 = NULL;

node\* curr1 = NULL;

node\* Insert(node\* NODE, int data) {

node\* curr = NULL;

if (head == NULL) {

NODE = new node;

NODE->data = data;

head = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

curr1 = NODE->next;

}

return NODE;

}

node\* Insert1(node\* NODE, int data) {

node\* curr = NULL;

if (head1 == NULL) {

NODE = new node;

NODE->data = data;

head1 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* merge\_lists(node\* NODE1) {

if (NODE1->next == NULL) {

NODE1->next = head1;

}

curr1 = NODE1->next;

return head;

}

void Sort() {

node\* temp = head;

node\* temp2 = head;

temp2 = temp2->next;

node\* Swap = NULL;

node\* check;

while (temp != NULL) {

while (temp2 != NULL) {

if (temp->data < temp2->data) {

swap(temp->data, temp2->data);

}

else {

temp2 = temp2->next;

}

}

temp = temp->next;

temp2 = temp;

if (temp2 == NULL) {

break;

}

temp2 = temp2->next;

}

}

void Print() {

node\* NODE = head;

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

int main() {

node\* Node = NULL;

node\* Node1 = NULL;

int l = 10;

while (true) {

cout << "1- To Input the Value in 1st Link list" << endl;

cout << "2- To Input the Value in 2nd Link list" << endl;

cout << "3- To Merge and sort" << endl;

cout << "4- To Print The Mereged Link List" << endl;

int data;

cin >> data;

if (data == 1) {

cout << "Enter The Data You Want to Insert" << endl;

int data = 0;

cin >> data;

Node = Insert(Node, data);

}

if (data == 2) {

cout << "Enter The Data You Want to Insert" << endl;

int data1 = 0;

cin >> data1;

Node1 = Insert1(Node1, data1);

}

if (data == 3) {

merge\_lists(Node);

Sort();

}

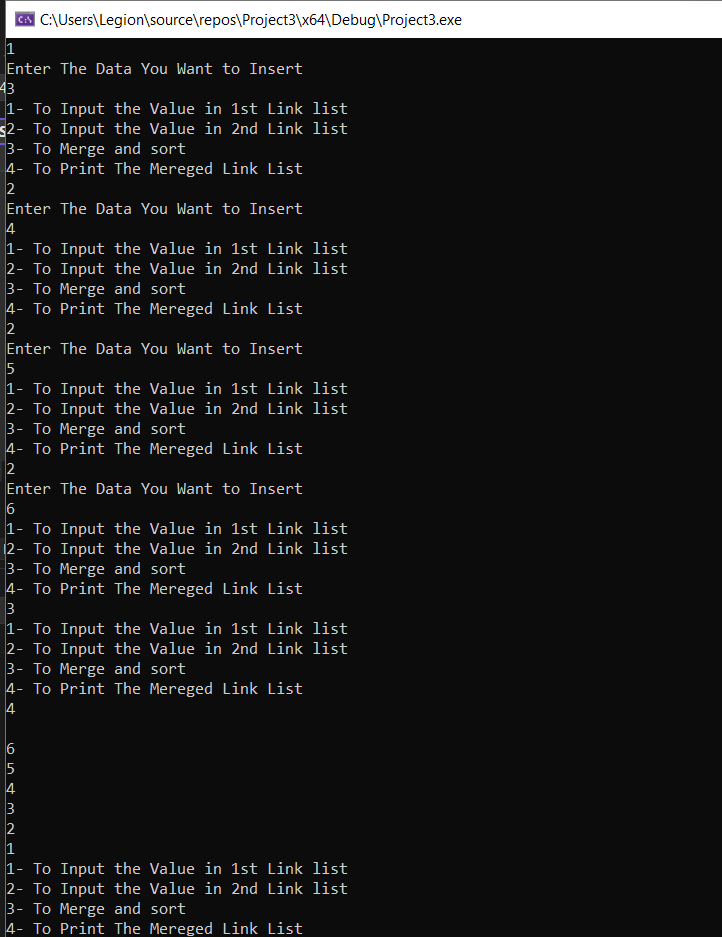
if (data == 4) {

Print();

}

}

}



Task 4:

#include<iostream>

using namespace std;

struct node {

int data;

node\* next;

};

node\* head = NULL;

node\* head1 = NULL;

node\* curr1 = NULL;

node\* head2 = NULL;

node\* Insert(node\* NODE, int data) {

node\* curr = NULL;

if (head == NULL) {

NODE = new node;

NODE->data = data;

head = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

curr1 = NODE;

}

return NODE;

}

node\* Insert1(node\* NODE, int data) {

node\* curr = NULL;

if (head1 == NULL) {

NODE = new node;

NODE->data = data;

head1 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* Insert3(node\* NODE, int data) {

node\* curr = NULL;

if (head2 == NULL) {

NODE = new node;

NODE->data = data;

head2 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* merge\_lists(node\* NODE1) {

node\* temp = head1;

int count = 0;

while (count < 3) {

if (count == 2) {

curr1->next = temp;

break;

}

else {

temp = temp->next;

count = count + 1;

}

}

return head;

}

void Print() {

node\* NODE = head;

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Common\_Element() {

head2 = NULL;

node\* NODE = head;

node\* NODE2 = NULL;

while (NODE != NULL) {

NODE2 = Insert3(NODE2, NODE->data);

NODE = NODE->next;

}

node\* temp2 = head2;

node\* temp3 = head2;

int count = 0;

int data = 0;

while (temp2 != NULL) {

temp3 = head2;

data = temp2->data;

while (temp3 != NULL) {

if (temp3->data == data) {

count = count + 1;

temp3->data = 0;

}

temp3 = temp3->next;

}

if (data != 0 && count > 1) {

cout << "Your Data : " << data << " Comes " << count << " Times";

cout << endl;

}

count = 0;

temp2 = temp2->next;

}

cout << endl;

}

void Print1() {

node\* NODE = head;

cout << endl;

while (NODE != curr1->next) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Print3() {

node\* NODE = head1;

cout << "Your Data : " << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

int main() {

node\* Node = NULL;

node\* Node1 = NULL;

int l = 10;

while (true) {

cout << "1- Input the Value in 1st Link list" << endl;

cout << "2- Input the Value in 2nd Link list" << endl;

cout << "3- Merge lists " << endl;

cout << "4- Print The Merged Link List" << endl;

cout << "5- Find Common Element" << endl;

cout << "6- Quit" << endl;

int data;

cin >> data;

if (data == 1) {

cout << "Enter The Data You Want to Insert" << endl;

int data = 0;

cin >> data;

Node = Insert(Node, data);

}

if (data == 2) {

cout << "Enter The Data You Want to Insert" << endl;

int data1 = 0;

cin >> data1;

Node1 = Insert1(Node1, data1);

}

if (data == 3) {

merge\_lists(Node1);

}

if (data == 4) {

Print();

}

if (data == 5) {

Common\_Element();

}

if (data == 6) {

break;

}

}

}



Task 5:

#include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* next;

struct Node\* prev;

};

void insert\_at\_begining(struct Node\*\* head, int new\_data) {

struct Node\* newNode = new Node;

newNode->data = new\_data;

newNode->next = (\*head);

newNode->prev = NULL;

if ((\*head) != NULL)

(\*head)->prev = newNode;

(\*head) = newNode;

}

void insert\_After(struct Node\* prev\_node, int new\_data) {

if (prev\_node == NULL) {

cout << "node cannot be NULL";

return;

}

struct Node\* newNode = new Node;

newNode->data = new\_data;

newNode->next = prev\_node->next;

prev\_node->next = newNode;

newNode->prev = prev\_node;

if (newNode->next != NULL)

newNode->next->prev = newNode;

}

void insert\_at\_end(struct Node\*\* head, int new\_data) {

struct Node\* newNode = new Node;

struct Node\* last = \*head;

newNode->data = new\_data;

newNode->next = NULL;

if (\*head == NULL) {

newNode->prev = NULL;

\*head = newNode;

return;

}

while (last->next != NULL)

last = last->next;

last->next = newNode;

newNode->prev = last;

return;

}

void Print(struct Node\* node) {

struct Node\* last;

while (node != NULL) {

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

last = node;

node = node->next;

}

if (node == NULL)

cout << "NULL";

}

Node\* delete\_first\_node(Node\* head){

if (head == NULL)

return NULL;

// Move the head pointer to the next node

Node\* temp = head;

head = head->next;

delete temp;

return head;

}

void delete\_middle(int value, Node\* head) {

Node\* Temp = head;

Node\* temp2 = NULL;

Node\* temp3 = NULL;

while (Temp != NULL) {

if (Temp->data == value) {

temp3 = Temp;

temp2 = Temp;

temp2 = Temp->prev;

temp3 = Temp->next;

if (temp3 == NULL) {

break;

}

delete Temp;

temp3->prev = temp2;

temp2->next = temp3;

Temp = temp3;

}

Temp = Temp->next;

}

};

Node\* delete\_last\_node( Node\* head){

if (head == NULL)

return NULL;

if (head->next == NULL) {

delete head;

return NULL;

}

Node\* second\_last = head;

while (second\_last->next->next != NULL)

second\_last = second\_last->next;

delete (second\_last->next);

second\_last->next = NULL;

return head;

}

int main() {

int choice, x;

struct Node\* head = NULL;

while (true) {

cout << endl;

cout << "1- Enter at first " << endl;

cout << "2- Enter at end " << endl;

cout << "3- Enter at position " << endl;

cout << "4 - Delete node in middle " << endl;

cout << "5- display" << endl;

cout << "6- Delete first node" << endl;

cout << "7- Delete last node" << endl;

cin >> choice;

if (choice == 1) {

cout << "enter number : ";

cin >> x;

insert\_at\_begining(&head, x);

}

if (choice == 2) {

cout << "enter number : ";

cin >> x;

insert\_at\_end(&head, x);

}

if (choice == 3) {

cout << "enter number : ";

cin >> x;

insert\_After(head->next, x);

}

if (choice == 4) {

cout << "enter the number you want to delete : ";

cin >> x;

delete\_middle(x, head);

}

if (choice == 5) {

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

Print(head);

}

if (choice == 6) {

head = delete\_first\_node(head);

}

if (choice == 7) {

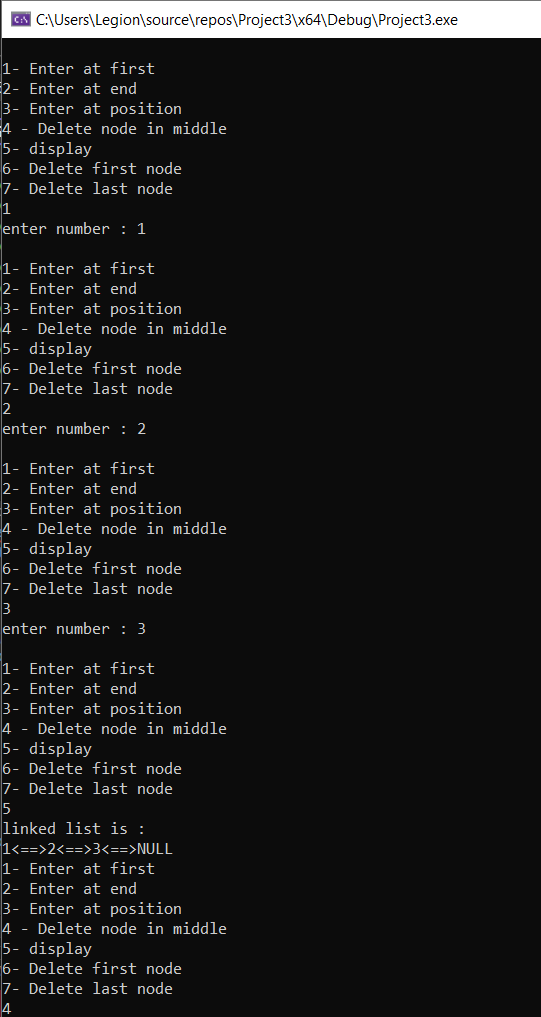
head = delete\_last\_node(head);

}

}

return 0;

}



Text

Description automatically generated

Text

Description automatically generated