1. Swap Every Two Nodes in a linked list.   
   
Given a linked list, swap every two nodes for example 1→2→3→4→5→6   
should   
become 2→1→4→3→6→5   
· Name this function swap Nodes()   
· Print the result!

#include <iostream>

using namespace std;

class list {

public:

int roll;

list\* next;

};

list\* head = NULL;

list\* create\_node() {

list\* newnode = new list;

return newnode;

}

void read\_node(list\* newnode) {

cout << "ENTER THE DATA : "; cin >> newnode->roll; cout << endl;

}

void insert\_at\_head(list\* newnode) { //ok

if (head == NULL) {

head = newnode;

head->next = NULL;

}

else {

newnode->next = head;

head = newnode;

}

}

void swap() {

list\* temp = head;

list\* temp\_next = head->next;

list\* temp\_next\_save = temp\_next->next;

temp\_next->next = temp;

temp->next = temp\_next\_save;

head = temp\_next; //updates head

list\* prev = temp; //saves prev

//loop for updating the rest

while (prev->next->next != NULL) { //makes sure to update two nodes at once

list\* first\_node = prev->next;

list\* second\_node = first\_node->next;

list\* second\_node\_next = second\_node->next;

prev->next = second\_node;

second\_node->next = first\_node;

first\_node->next = second\_node\_next;

prev = prev->next->next;

first\_node = first\_node->next->next;

second\_node = second\_node->next->next;

}

}

void display(){

list\* temp = head;

if (head == NULL) {

cout << "NOTHING IN THE LIST TO BE DISPLAYED" << endl;

}

else {

while (temp != NULL) {

cout << temp->roll << endl;

temp = temp->next;

}

}

}

int main() {

list l;

while (1) {

int opt;

cout << "ENTER 1 TO INPUT DATA" << endl;

cout << "ENTER 2 TO SWAP THE NODES" << endl;

cout << "ENTER 3 TO DISPLAY" << endl;

cout << "ENTER 4 TO EXIT" << endl;

cin >> opt; cout << endl;

switch (opt) {

case 1:

{

list\* newnode = create\_node();

read\_node(newnode);

insert\_at\_head(newnode);

break;

}

case 2:

{

swap();

break;

}

case 3:

{

display();

break;

}

case 4:

{

exit(0);

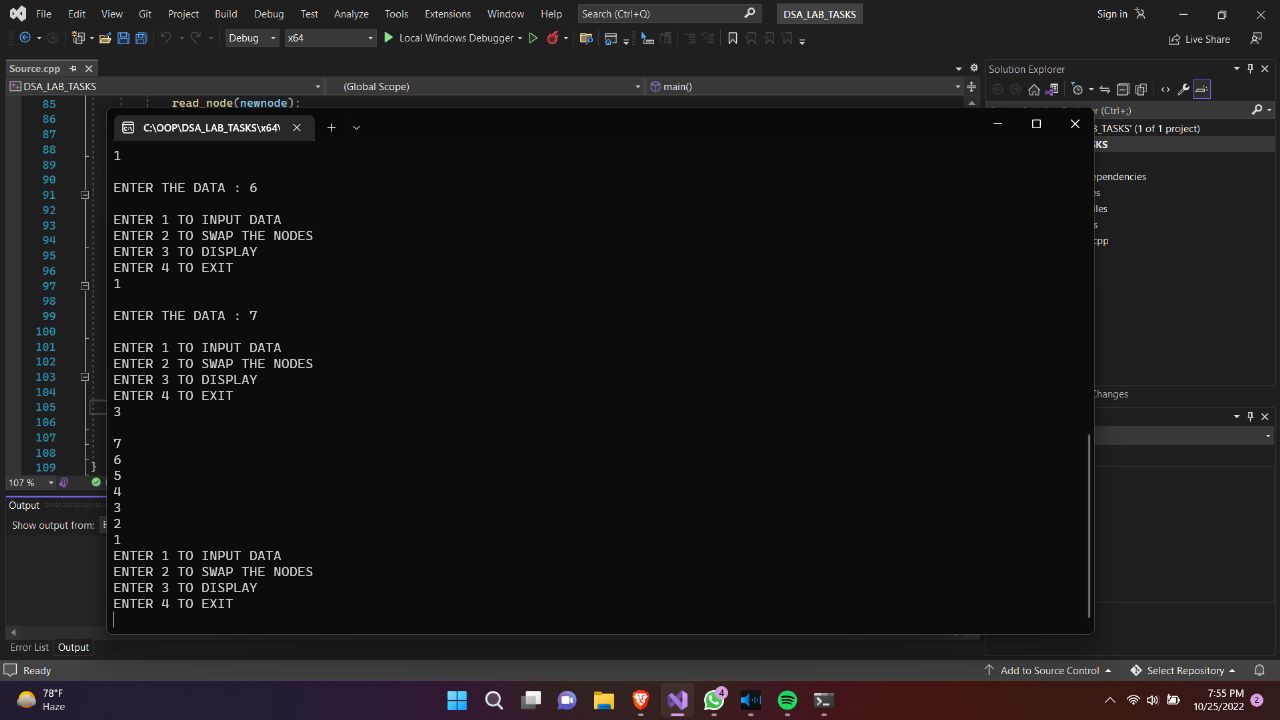
break;

}

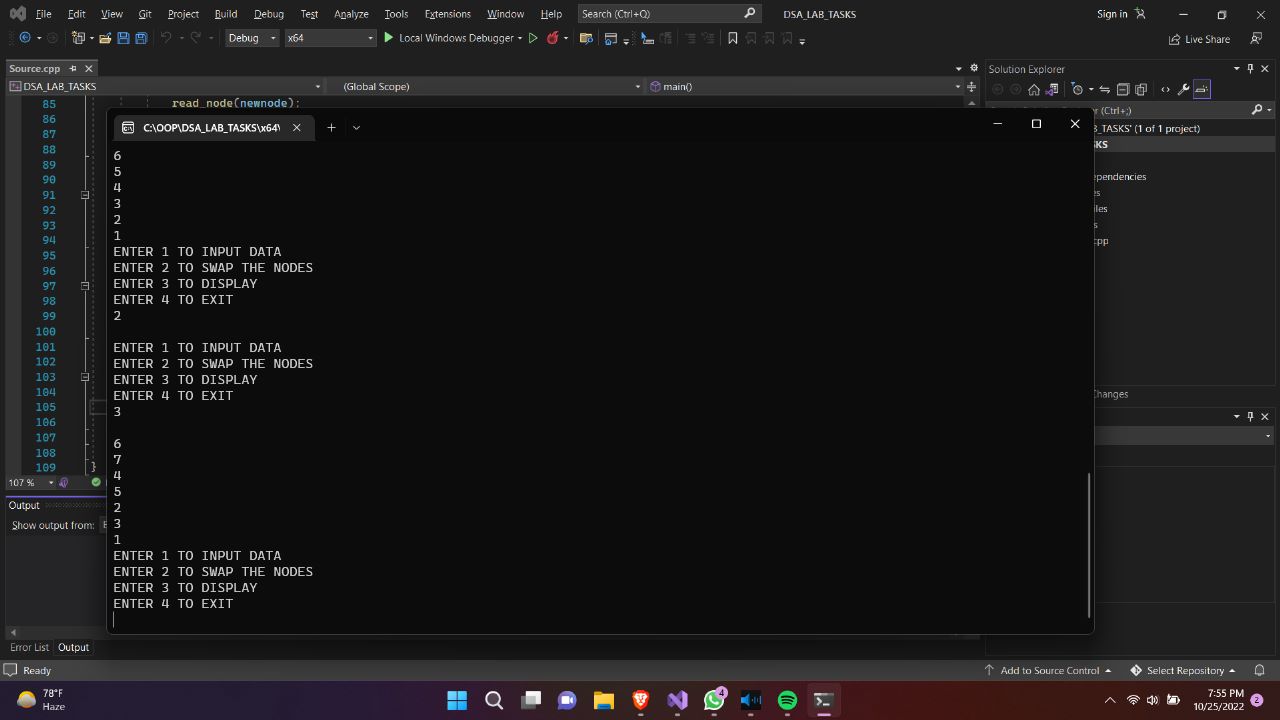
}

}

}



AFTER SWAP :



2. Merge a linked list into another linked list at alternate positions.  
 Given two linked lists, insert nodes of second list into first list at alternate   
positions of   
first list. For example: 1→5→7→3→9 and 6→10→2→4 are two lists. So, the   
first list   
should become 1→6→5→10→7→2→3→4→9 and second list should   
become empty.   
The nodes of second list should only be inserted when there are positions   
available.   
If the first list is:  
   
1→2→3 and   
second list is: 4→5→6→7   
then first list should become: 1→4→2→5→3→6 and   
second list: 7   
   
Name this function: mergeListAlternatively().

#include <iostream>

using namespace std;

class lists {

public:

int roll;

lists\* head1 = NULL;

lists\* head2 = NULL;

lists\* next1;

lists\* next2;

lists\* create\_node() {

lists\* newnode = new lists;

return newnode;

}

void read\_data(lists\* newnode) {

cout << "-> "; cin >> newnode->roll; cout << endl;

system("CLS");

}

void insert\_at\_head\_1(lists\* newnode) {

if (head1 == NULL) {

head1 = newnode;

newnode->next1 = NULL;

}

else {

newnode->next1 = head1;

head1 = newnode;

}

}

void insert\_at\_head\_2(lists\* newnode) {

if (head2 == NULL) {

head2 = newnode;

newnode->next2 = NULL;

}

else {

newnode->next2 = head2;

head2 = newnode;

}

}

void display1(lists\* head) {

lists\* temp = head;

if (head == NULL) {

cout << "\*\*NOTHING TO DISPLAY\*\*"<<endl;

}

else {

while (temp != NULL) {

cout << temp->roll << endl;

temp = temp->next1;

}

}

}

void display2(lists\* head) {

lists\* temp = head;

if (head == NULL) {

cout << "\*\*NOTHING TO DISPLAY\*\*" << endl;

}

else {

while (temp != NULL) {

cout << temp->roll << endl;

temp = temp->next2;

}

}

}

void alternate(lists\* h2) {

lists\* temp1 = head1;

lists\* temp2 = h2;

lists\* prev = NULL;

lists\* temp1\_next = temp1->next1; // 1 step ahead

temp1->next1 = temp2;

temp2->next1 = temp1\_next;

lists\* temp\_imp = NULL;

//WORKS

while (temp1\_next->next1 != NULL) {

lists\* temp\_next\_add = temp1\_next->next1;

temp1\_next->next1 = temp2->next2;

temp1\_next->next1->next1 = temp\_next\_add;

temp1\_next = temp1\_next->next1->next1;

temp\_imp = temp1\_next;

}

/\*if (temp\_imp->next1 == NULL) {

while (temp2->next2 != NULL) {

temp2 = temp2->next2;

}

temp\_imp->next1 = temp2;

temp2->next1 = NULL;

}\*/

}

};

int main() {

lists l;

int opt = 0;

while (1) {

cout << "ENTER 1 TO INPUT DATA FOR LIST 1" << endl;

cout << "ENTER 2 TO INPUT DATA FOR LIST 2" << endl;

cout << "ENTER 3 TO INPUT DATA OF LIST 2 IN LIST 1 AT ALTERNATE" << endl;

cout << "ENTER 4 TO DISPLAY LIST 1" << endl;

cout << "ENTER 5 TO DISPLAY LIST 2[FOR INP CHECK]" << endl;

cout << "ENTER 6 TO EXIT" << endl;

cin >> opt; cout << endl;

switch (opt) {

case 1:

{

system("CLS");

lists\* newnode = l.create\_node();

cout << "ENTER DATA FOR LIST 1 : - ";

l.read\_data(newnode);

l.insert\_at\_head\_1(newnode);

break;

}

case 2:

{

system("CLS");

lists\* newnode = l.create\_node();

cout << "ENTER DATA FOR LIST 2 : - ";

l.read\_data(newnode);

l.insert\_at\_head\_2(newnode);

break;

}

case 3:

{

lists\* h2\_copy = l.head2;

system("CLS");

l.alternate(h2\_copy);

break;

}

case 4:

{

system("CLS");

l.display1(l.head1);

break;

}

case 5:

{

system("CLS");

l.display2(l.head2);

break;

}

case 6:

{

exit(0);

break;

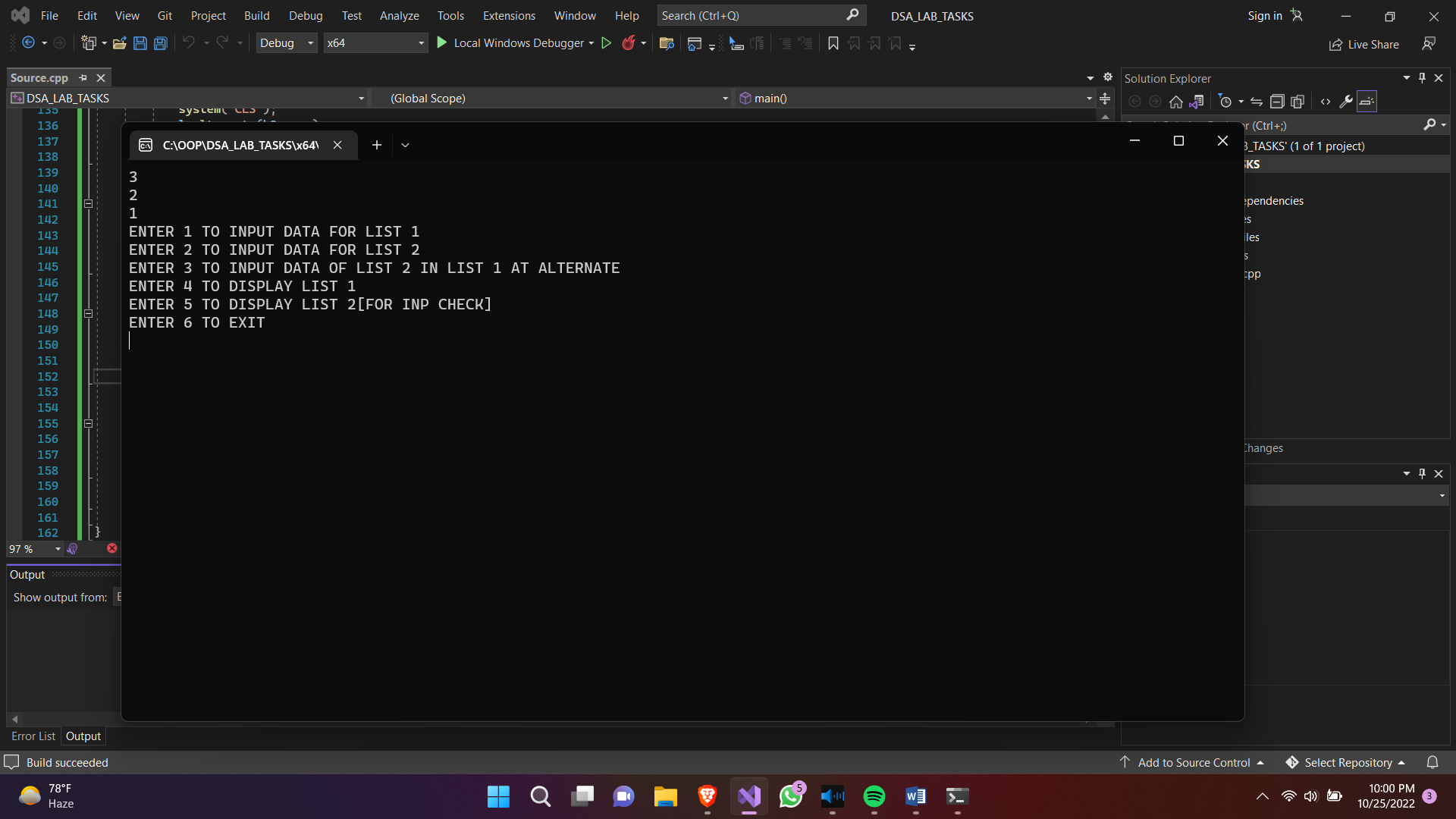
}

}

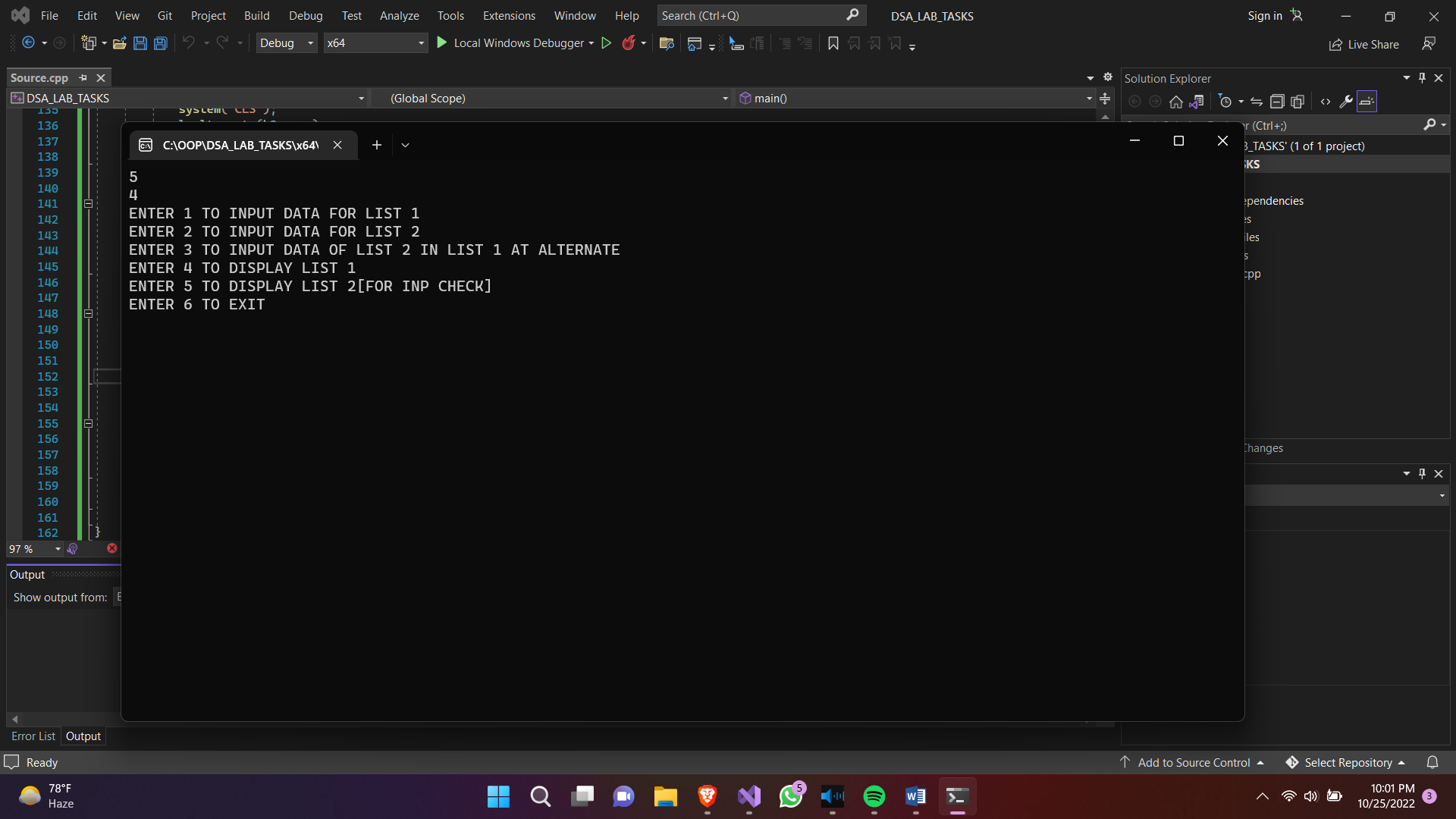
}

}

LIST NO 1:



LIST NO 2:



JOINED:

