1. Write program to Implement a stack using   
   a. singly linked list.

#include <iostream>

using namespace std;

class node {

public:

node\* next;

int roll;

};

node\* top = NULL;

class list {

public:

node\* create\_node() {

node\* newnode = new node;

return newnode;

}

void read\_data(node\* newnode) {

cout << "ENTER THE ROLL NO : "; cin >> newnode->roll; cout << endl;

}

void push(node\* newnode) {

if (top == NULL) {

top = newnode;

top->next = NULL;

}

else {

newnode->next = top;

top = newnode;

}

}

void pop() {

if (top == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

node\* temp = top;

top = top->next;

delete(temp);

}

}

bool is\_empty() {

if (top == NULL) {

return true;

}

else {

return false;

}

}

void display() {

if (top == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

node\* temp = top;

while (temp != NULL) {

cout << temp->roll; cout << endl;

temp = temp->next;

}

}

}

};

int main() {

int opt; list l;

while (1) {

cout << "ENTER 1 TO PUSH" << endl;

cout << "ENTER 2 TO POP" << endl;

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

cin >> opt; cout << endl;

switch (opt) {

case 1:

{

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

l.read\_data(newnode);

l.push(newnode);

break;

}

case 2:

{

l.pop();

break;

}

case 3:

{

l.display();

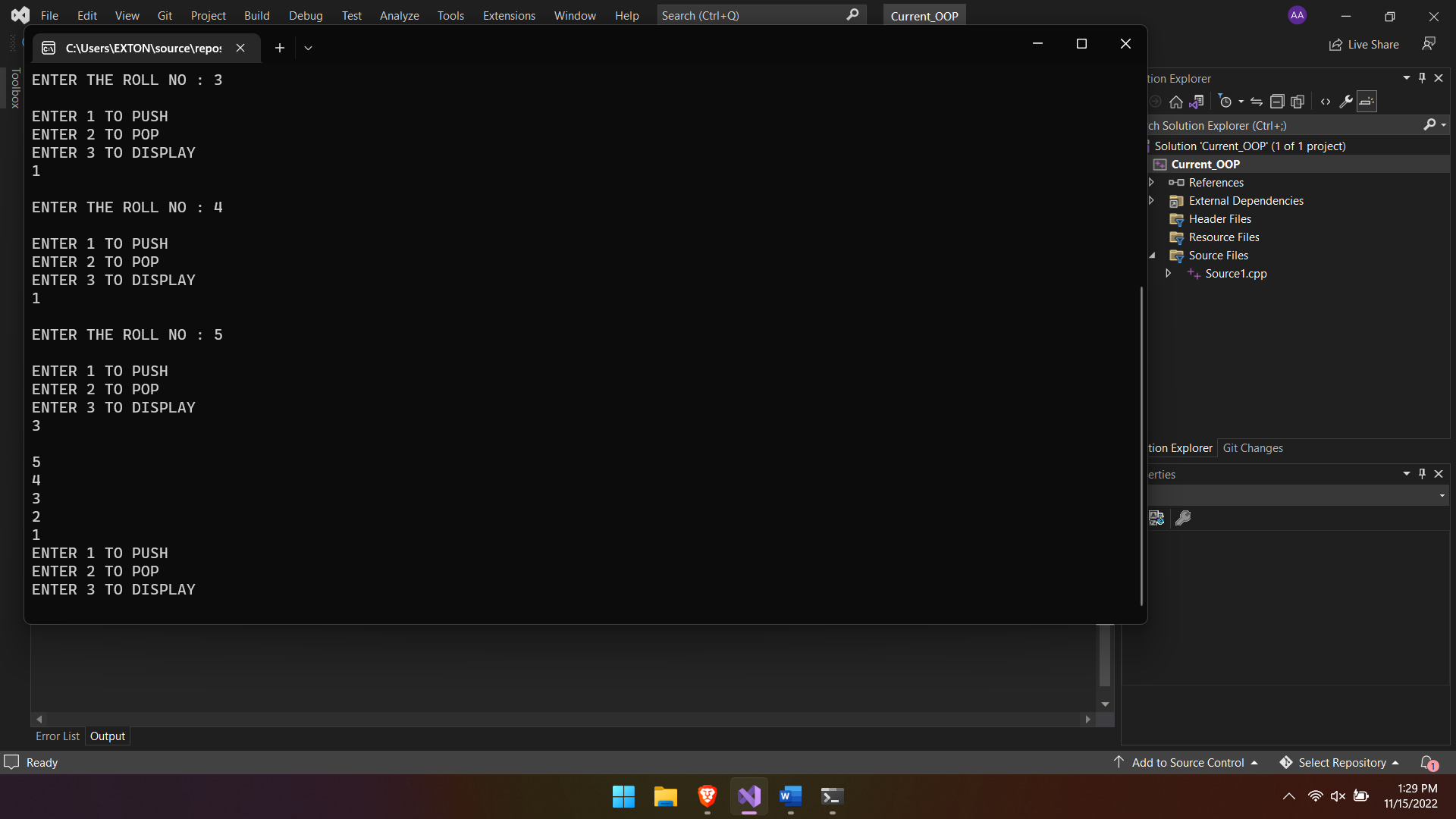
break;

}

}

}

}



2. Sort the stack using another temporary stack.  
Like if values in Stack S and in Stack T at the start are:

//push done

//pop done

//initial push done

#include <iostream>

using namespace std;

class node {

public:

node\* next;

int data;

};

node\* top\_final;

node\* top\_1; //temp

node\* top\_2; //temp

node\* create\_node() {

node\* newnode = new node;

return newnode;

}

void push\_top\_1(int i) {

node\* newnode = create\_node();

newnode->data = i;

if (top\_1 == NULL) {

top\_1 = newnode;

top\_1->next = NULL;

}

else {

newnode->next = top\_1;

top\_1 = newnode;

}

}

void push\_top\_2(int i) {

node\* newnode = create\_node();

newnode->data = i;

if (top\_2 == NULL) {

top\_2 = newnode;

top\_2->next = NULL;

}

else {

newnode->next = top\_2;

top\_2 = newnode;

}

}

void push\_top\_final(int i) {

node\* newnode = create\_node();

newnode->data = i;

if (top\_final == NULL) {

top\_final = newnode;

top\_final->next = NULL;

}

else {

newnode->next = top\_final;

top\_final = newnode;

}

}

void pop\_top\_1() {

if (top\_1 == NULL) {

top\_1 = NULL;

cout << "" << endl;

}

else {

node\* temp\_top\_1 = top\_1;

top\_1 = NULL;

}

}

void pop\_top\_2() {

if (top\_2 == NULL) {

top\_2 = NULL;

cout << "" << endl;

}

else {

node\* temp\_top\_2 = top\_2;

top\_2 = NULL;

}

}

//IMPORTANT

bool status\_temp\_1() {

if (top\_1 == NULL) {

return 1;

}

else {

return 0;

}

}

bool status\_temp\_2() {

if (top\_2 == NULL) {

return 1;

}

else {

return 0;

}

}

void ascend() {

//temp stack no 1

if (top\_1 == NULL) {

node\* temp\_top\_2 = top\_2;

node\* min\_node = top\_2;

int min\_data\_2 = top\_2->data;

while (temp\_top\_2 != NULL) {

if (min\_data\_2 > temp\_top\_2->data) {

min\_data\_2 = temp\_top\_2->data;

min\_node = temp\_top\_2;

}

else {

temp\_top\_2 = temp\_top\_2->next;

}

}

push\_top\_final(min\_data\_2); //pushes min value to final

node\* temp\_top = top\_2;

while (temp\_top->next != NULL) {

if (temp\_top == min\_node) {

node\* temp\_del = temp\_top;

temp\_top = temp\_top->next;

}

else {

push\_top\_1(temp\_top->data);

node\* temp\_pop = temp\_top;

temp\_top = temp\_top->next;

pop\_top\_2();

}

}

if (temp\_top->next == NULL) {

if (temp\_top == min\_node) {

node\* temp\_del = temp\_top;

temp\_top = temp\_top->next;

pop\_top\_2();

}

else {

push\_top\_1(temp\_top->data);

pop\_top\_2();

}

}

}

//temp stack no 2

else if (top\_2 == NULL) {

node\* temp\_top\_1 = top\_1;

node\* min\_node = top\_1;

int min\_data\_1 = top\_1->data;

while (temp\_top\_1 != NULL) {

if (min\_data\_1 > temp\_top\_1->data) {

min\_data\_1 = temp\_top\_1->data;

min\_node = temp\_top\_1;

}

else {

temp\_top\_1 = temp\_top\_1->next;

}

}

push\_top\_final(min\_data\_1); //pushes min value to final

node\* temp\_top = top\_1;

while (temp\_top->next != NULL) {

if (temp\_top == min\_node) {

node\* temp\_del = temp\_top;

temp\_top = temp\_top->next;

delete(temp\_del);

}

else {

push\_top\_2(temp\_top->data);

node\* temp\_pop = temp\_top;

temp\_top = temp\_top->next;

pop\_top\_1();

}

}

if (temp\_top->next == NULL) {

if (temp\_top == min\_node) {

node\* temp\_del = temp\_top;

temp\_top = temp\_top->next;

pop\_top\_1();

}

else {

push\_top\_2(temp\_top->data);

pop\_top\_1();

}

}

}

}

void display\_final() {

node\* temp\_final = top\_final;

while (temp\_final != NULL) {

cout << temp\_final->data << endl;

temp\_final = temp\_final->next;

}

}

int main() {

//initial values

push\_top\_1(1);

push\_top\_1(5);

push\_top\_1(2);

push\_top\_1(8);

push\_top\_1(9);

//ascend

int count = 5;

int i = 0;

while (i<5) {

ascend();

i++;

}

display\_final();

}

INITIAL INPUT:

1

5

2

8

9

OUTPUT:

