**Ahtisham Ahmed**

**014-BSCS**

**TASK 1:**

#include <iostream>

using namespace std;

class node {

public:

node\* next;

int data;

static int count;

};

node\* head = NULL;

int node::count = 0;

class list :public node {

public:

node\* create\_node() {

node\* newnode = new node;

return newnode;

}

void read\_data(node& newnode) {

cout << "Enter the ID : ";

cin >> newnode.data;

cout << endl;

}

void insert\_at\_head() {

node\* newnode = create\_node();

read\_data(\*newnode);

if (head == NULL) {

head = newnode;

head->next = NULL;

count++;

}

else {

newnode->next = head;

head = newnode;

count++;

}

}

void display() {

if (head == NULL) {

cout << "The list is empty" << endl;

}

else {

node\* temp = head;

while (temp != NULL) {

cout << temp->data << "\t";

temp = temp->next;

}

cout << endl;

}

}

void delete\_even() {

int temp\_count = 1;

if (head == NULL) {

cout << "the list is empty" << endl;

}

else {

node\* prev = NULL;

node\* curr = head;

while (temp\_count <= count && curr != NULL) {

if (temp\_count % 2 == 0) {

prev->next = curr->next;

//count--;

}

prev = curr;

curr = curr->next;

temp\_count++;

}

}

}

void delete\_odd() {

int temp\_count = 1;

if (head == NULL) {

cout << "the list is empty" << endl;

}

else {

node\* prev = NULL;

node\* curr = head;

while (temp\_count <= count && curr != NULL) {

if (temp\_count == 1) {

head = head->next;

}

else if (temp\_count == count && temp\_count % 2 != 0) {

prev->next = NULL;

}

else if (temp\_count % 2 != 0) {

prev->next = curr->next;

//count--;

}

prev = curr;

curr = curr->next;

temp\_count++;

}

}

}

};

int main() {

int opt;

list l;

while (1) {

cout << "Enter 0 to delete even positioned nodes" << endl;

cout << "Enter 1 to delete odd positioned nodes" << endl;

cout << "Enter 2 to enter data to the list" << endl;

cout << "Enter 3 to Display the list" << endl;

cin >> opt;

cout << endl;

switch (opt) {

case 0:

{

l.delete\_even();

break;

}

case 1:

{

l.delete\_odd();

break;

}

case 2:

{

l.insert\_at\_head();

break;

}

case 3:

{

l.display();

break;

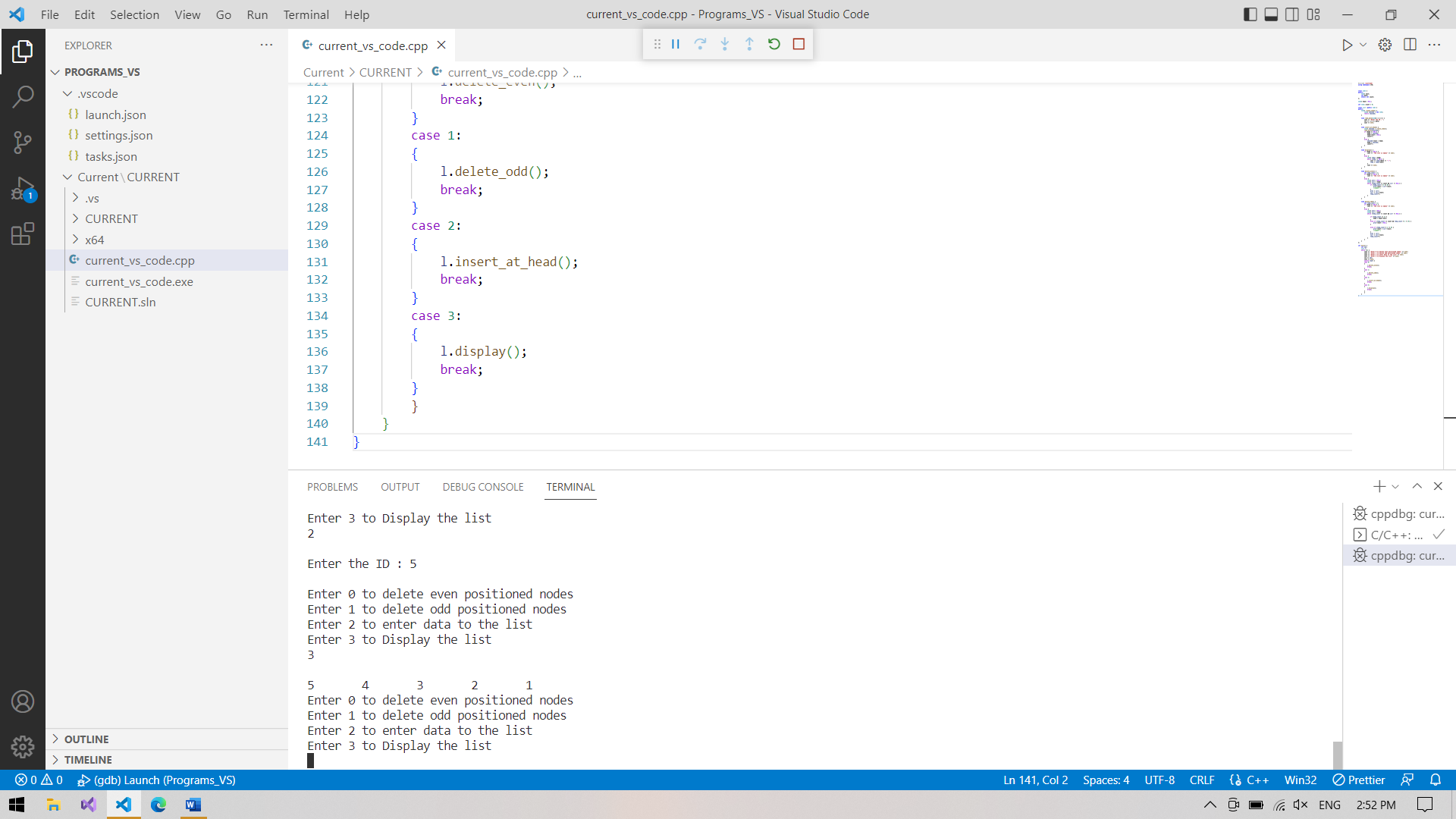
}

}

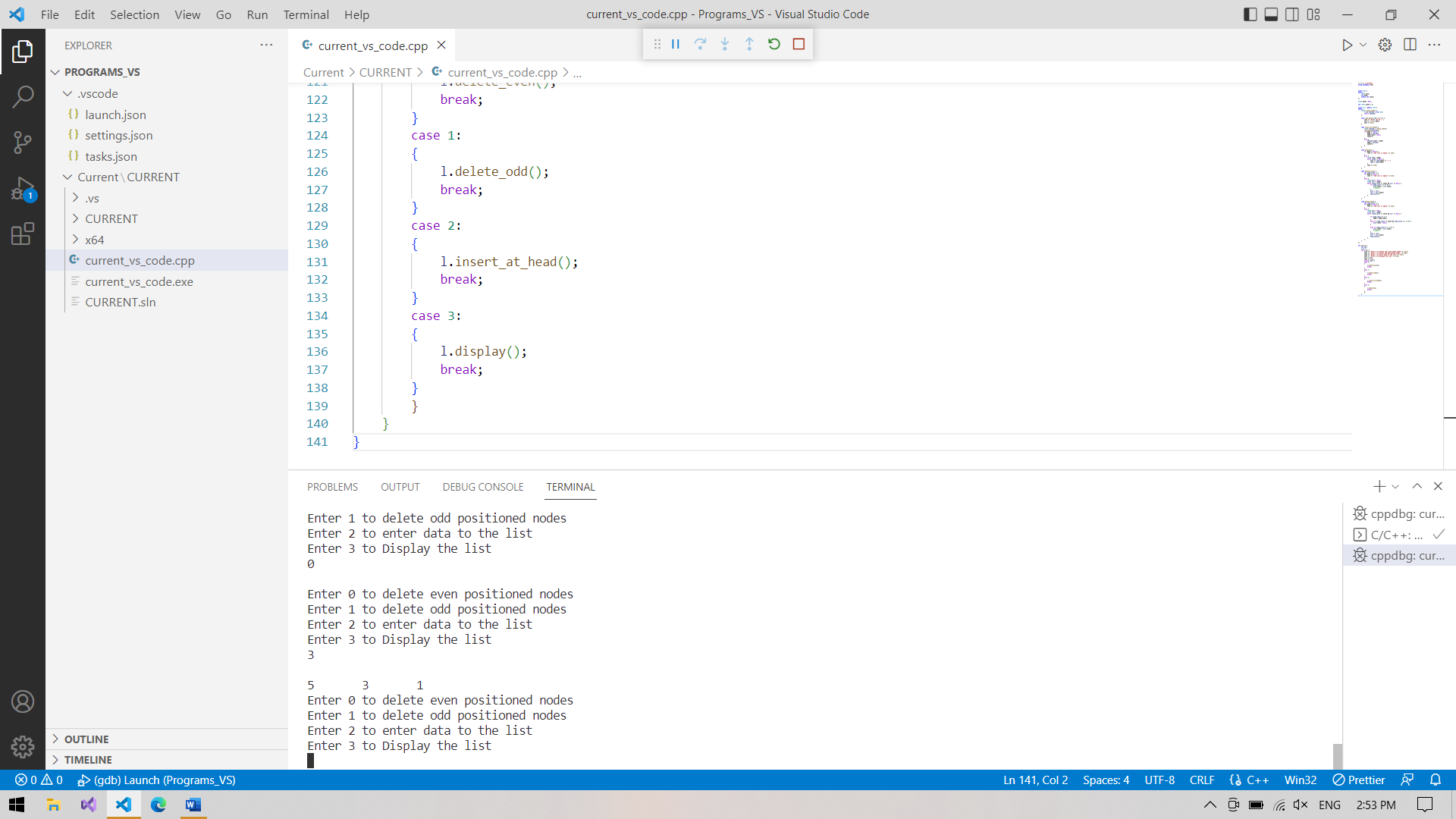
}

}

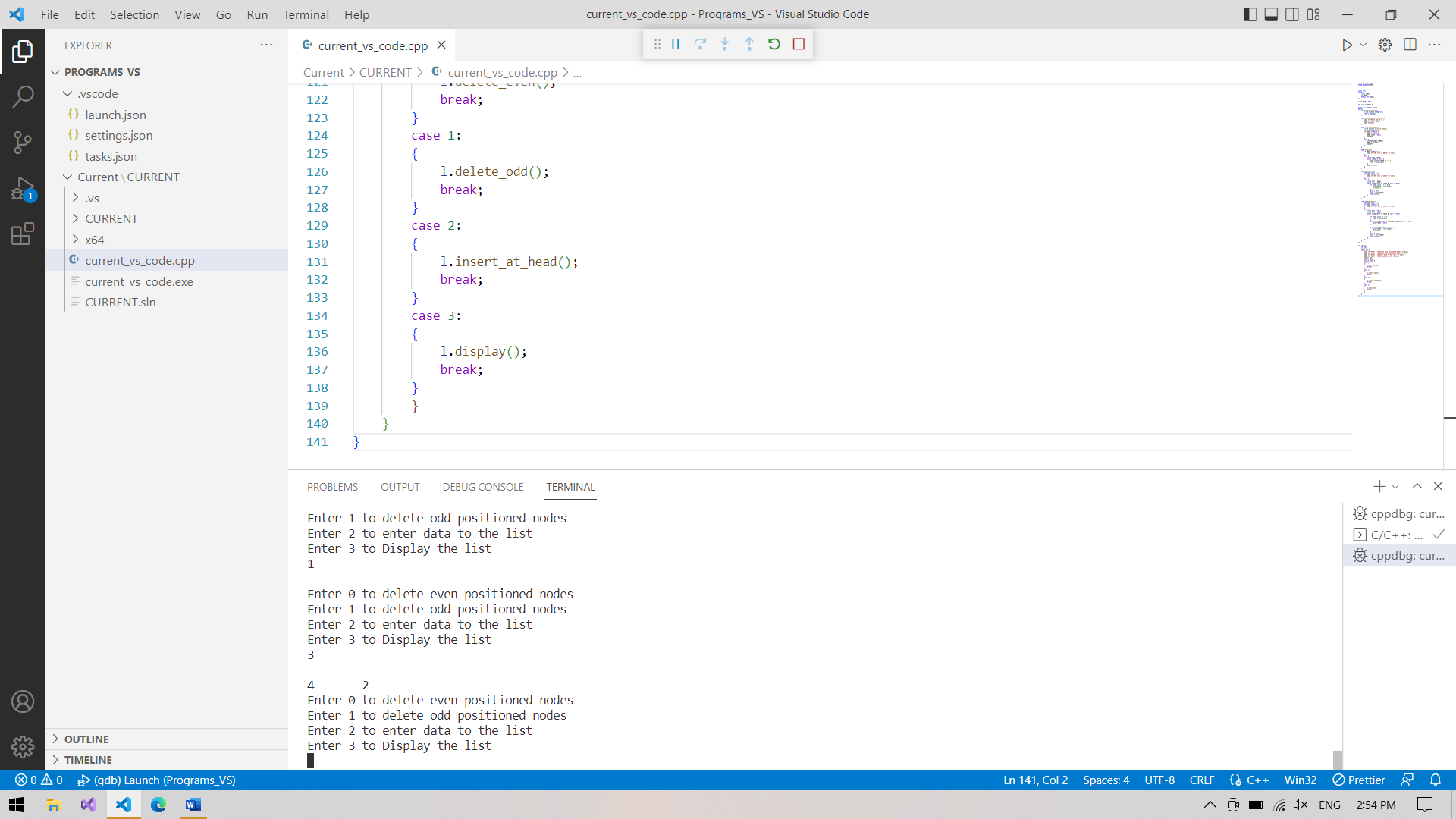
ORIGINAL LIST :



EVEN DELETION :



ODD DELETION :



**TASK 2:**

#include <iostream>

using namespace std;

class original\_bag\_stack {

public:

    int arr[5];

    static int top\_original;

    void insert() {

        if (top\_original == 4) {

            cout << "The Bag is Full" << endl;

        }

        else {

            top\_original++;

            int data\_inserted;

            cout << "Enter the integer : ";

            cin >> data\_inserted;

            cout << endl;

            arr[top\_original] = data\_inserted;

        }

    }

    void display\_original() {

        if (top\_original == -1) {

            cout << "The original bag is empty" << endl;

        }

        else {

            int temp\_trav = top\_original;

            while (temp\_trav >= 0) {

                cout << arr[temp\_trav] << "\t";

                temp\_trav--;

            }

            cout << endl;

        }

    }

    int size\_extra() {

        return top\_original;

    }

    void moved\_data() {

        top\_original = -1;

    }

};

int original\_bag\_stack::top\_original = -1;

class extra\_bag\_stack {

public:

    int arr[5];

    static int top\_extra;

    void insert\_extra(int arr\_passed[], int size\_original) {

        if (top\_extra == 4) {

            cout << "The Bag is Full" << endl;

        }

        else {

            int i = size\_original;

            if (i == 0) {

                top\_extra++;

                arr[top\_extra] = arr\_passed[0];

            }

            else{

                while (i >= 0) {

                    top\_extra++;

                    arr[top\_extra] = arr\_passed[i];

                    i--;

                }

            }

        }

    }

    void display\_extra() {

        if (top\_extra == -1) {

            cout << "The extra bag is empty" << endl;

        }

        else {

            int temp\_trav = top\_extra;

            while (temp\_trav >= 0) {

                cout << arr[temp\_trav] << "\t";

                temp\_trav--;

            }

            cout << endl;

        }

    }

    void remove\_top() {

        if (top\_extra == -1) {

            cout << "The extra bag is empty" << endl;

        }

        else {

            top\_extra--;

        }

    }

};

int extra\_bag\_stack::top\_extra = -1;

int main() {

    int opt;

    original\_bag\_stack o;

    extra\_bag\_stack e;

    while (1) {

        cout << "Enter 1 to enter data" << endl;

        cout << "Enter 2 to move data to other bag" << endl;

        cout << "Enter 3 to remove data on top of the other bag" << endl;

        cout << "Enter 4 to display data in original bag" << endl;

        cout << "Enter 5 to display data in extra bag" << endl;

        cin >> opt;

        cout << endl;

        switch (opt) {

        case 1:

        {

            o.insert();

            break;

        }

        case 2:

        {

            e.insert\_extra(o.arr, o.size\_extra());

            o.moved\_data();

            break;

        }

        case 3:

        {

            e.remove\_top();

            break;

        }

        case 4:

        {

            o.display\_original();

            break;

        }

        case 5:

        {

            e.display\_extra();

            break;

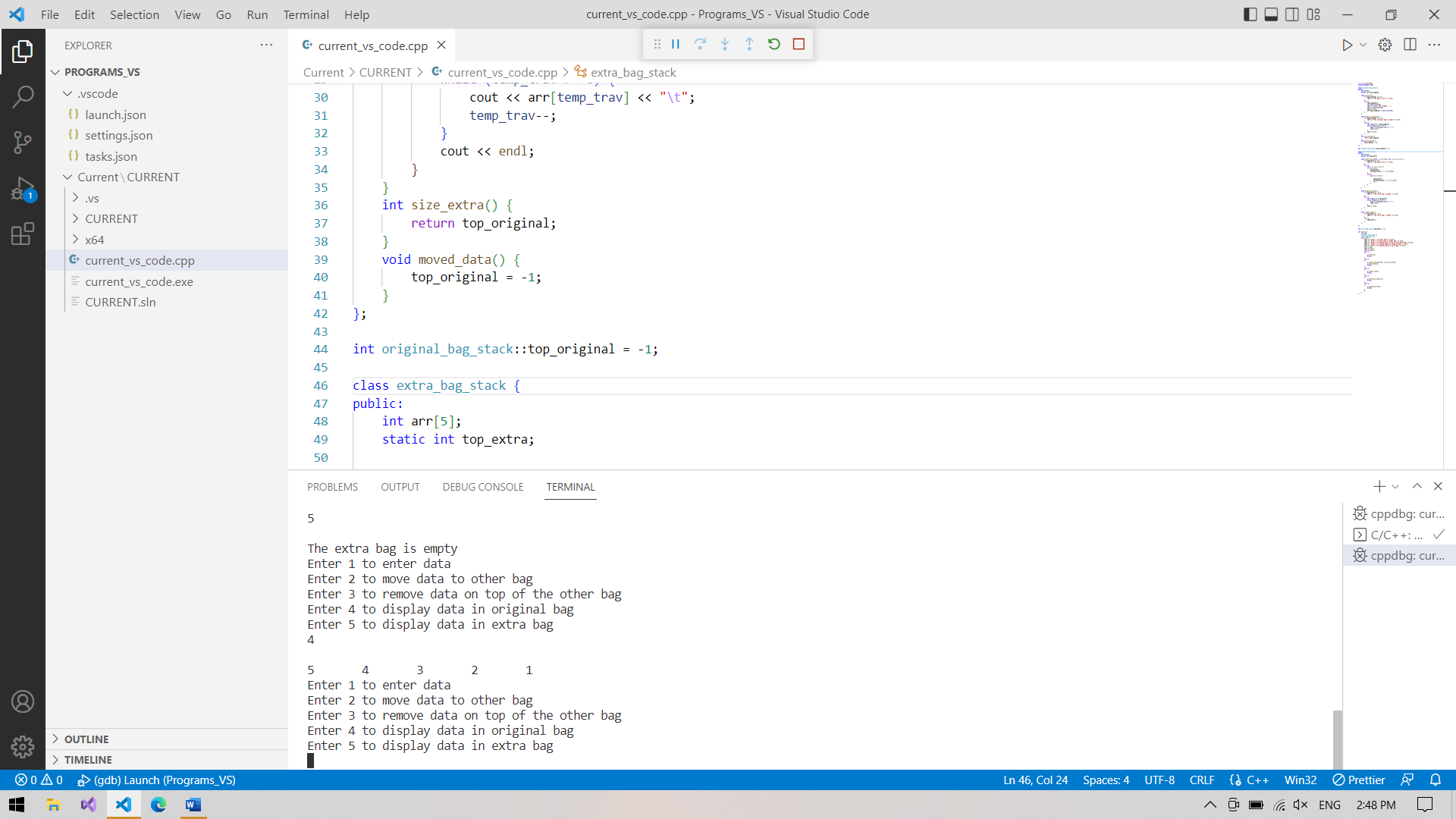
        }

        }

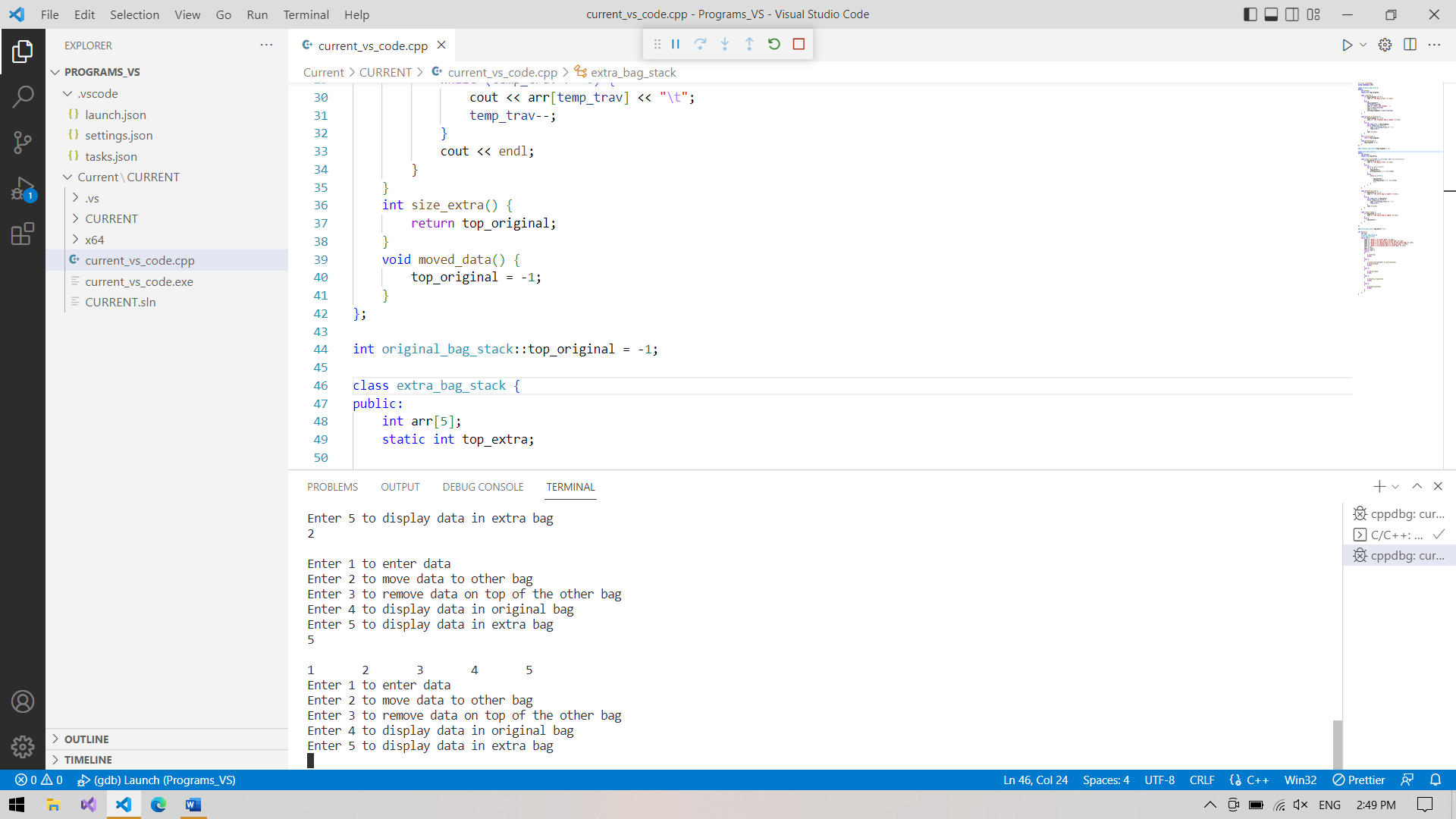
    }

}

ORIGINAL BAG :



EXTRA BAG:



AFTER DELETION:

