**Global Trend Programming Profile Assessment Questions With Solutions**

1. Write a function to reverse a singly linked list.The function should take the head of the list and return the new head of the reversed list.

Function reverse singly Linked list:  
  
Node\* reverseList(Node\* head) {

    Node\* pre=NULL;

    Node\* cure=head;

    Node\* forward=NULL;

    while(cure!=NULL){

        forward=cure->next;

        cure->next=pre;

        pre=cure;

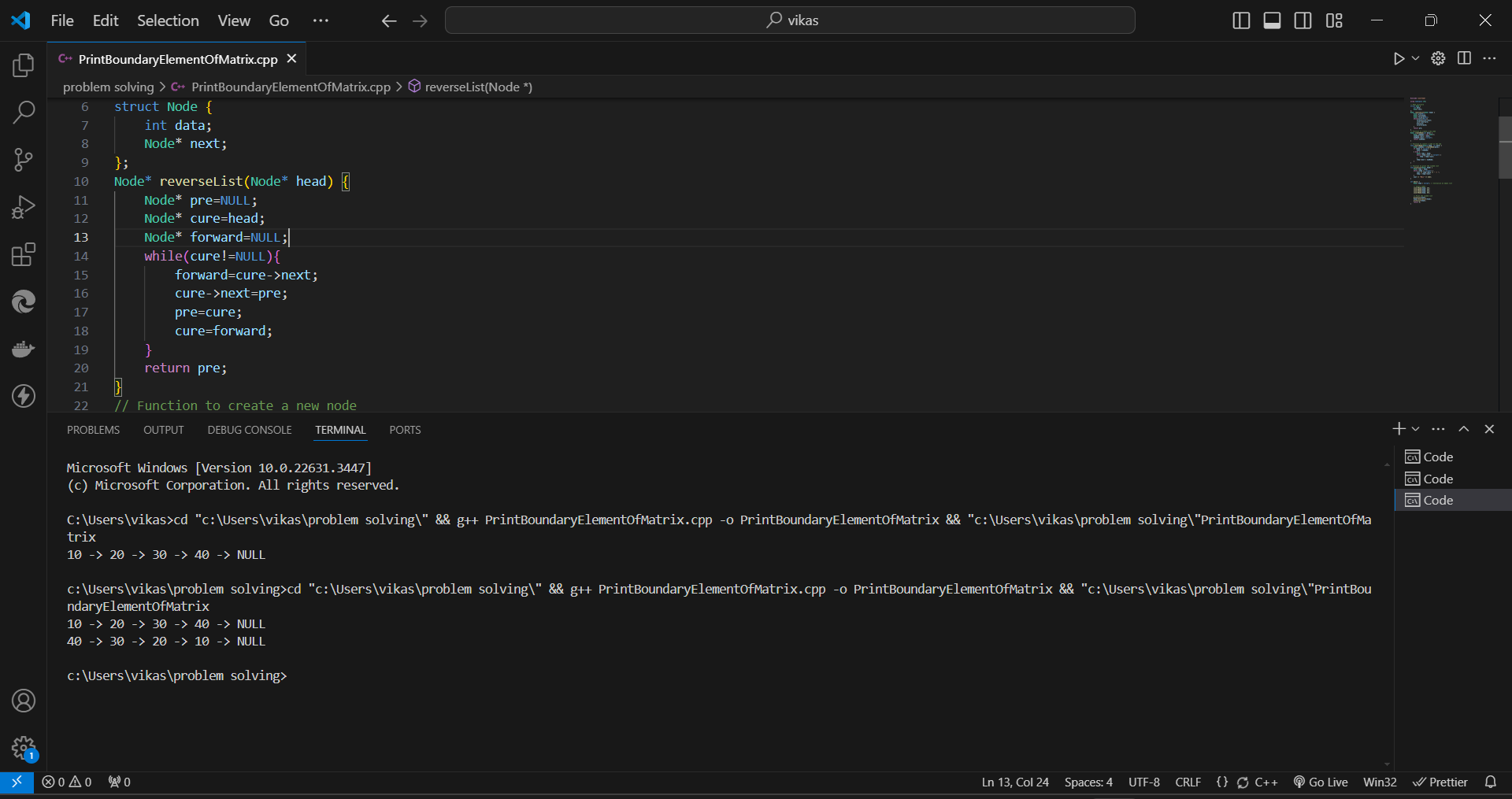
        cure=forward;

    }

    return pre;

}

Output:



1. Given a string, find the length of the longest substring without repeating characters.The function should return an integer representing the length of the longest substring without repeating characters.

Function for longest substring  
  
int LongestSubstringWithoutRepeatingCharacter(string s) {

// Below is the Vector to store the last index of each character (256 for extended ASCII)  
 // here I assume all type of character.

vector<int>lastIndex(256, -1);

int maxLength=0;

int start=0;

for (int i=0;i<s.length();i++) {

if (lastIndex[s[i]]!=-1) {

start=max(start,lastIndex[s[i]]+1);

}

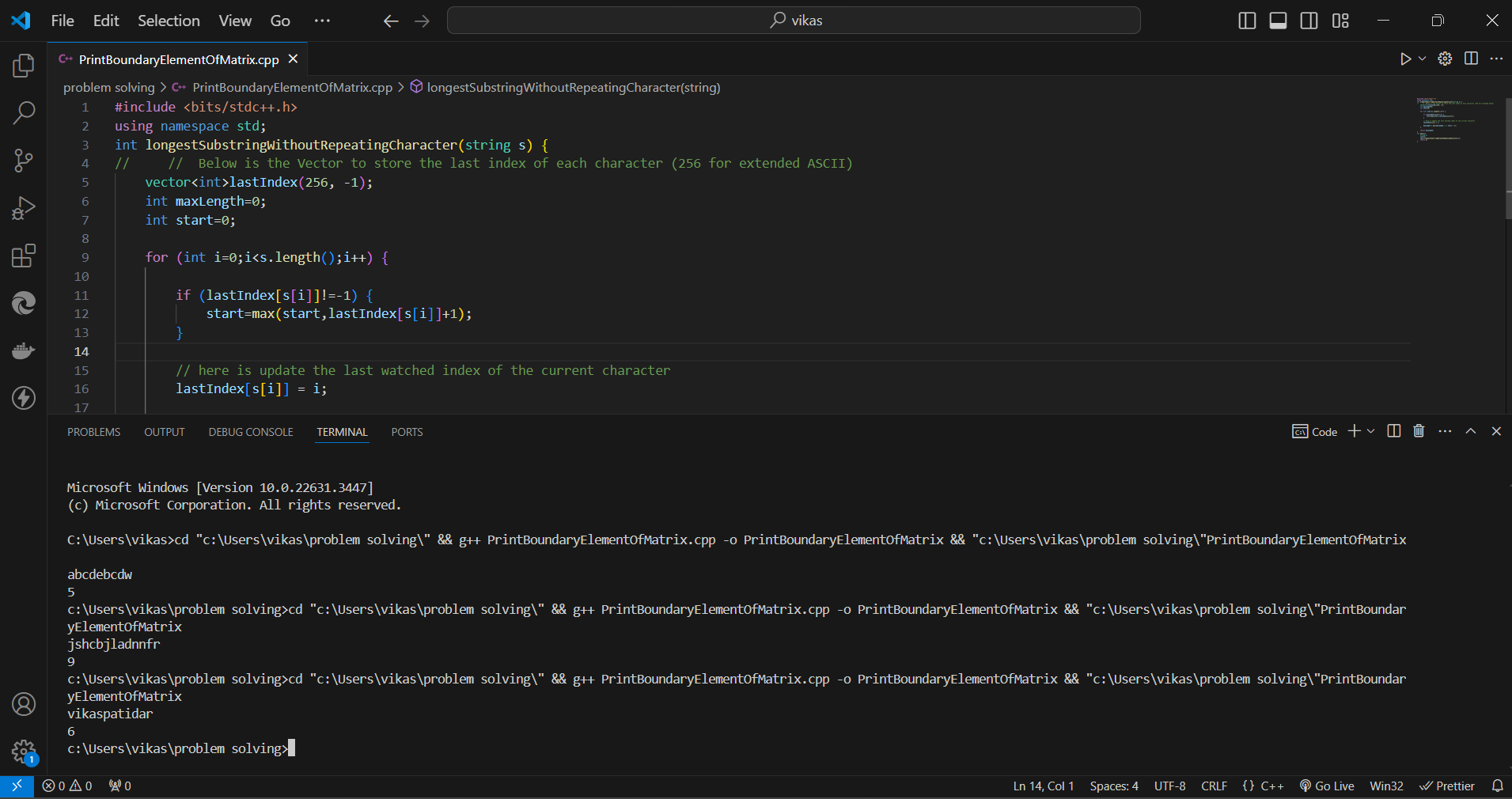
// here is update the last watched index of the current character

lastIndex[s[end]] = i;

maxLength = max(maxLength, i - start + 1);

}

return maxLength;

}  
  
  
Output:  


1. Given a non-empty binary tree, find the maximum path sum. A path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain at least one node and does not need to go through the root.The function should return an integer representing the maximum path sum.

Function for maximum sum path  
  
int postOrderTraverse(TreeNode\* root,int &ans){

if(root==NULL){

return 0;

}

int left=postOrderTraverse(root->left,ans);

int right=postOrderTraverse(root->right,ans);

if(left+root->data <root->data && right+root->data <root->data && root->data+left+right){

ans=max(ans,root->data);

return root->data;

}

int a=INT\_MIN;

int b=left+right+root->data;

if(left>right){

a=left+root->data;

}else{

a=right+root->data;

}

ans=max(ans,max(a,b));

return a;

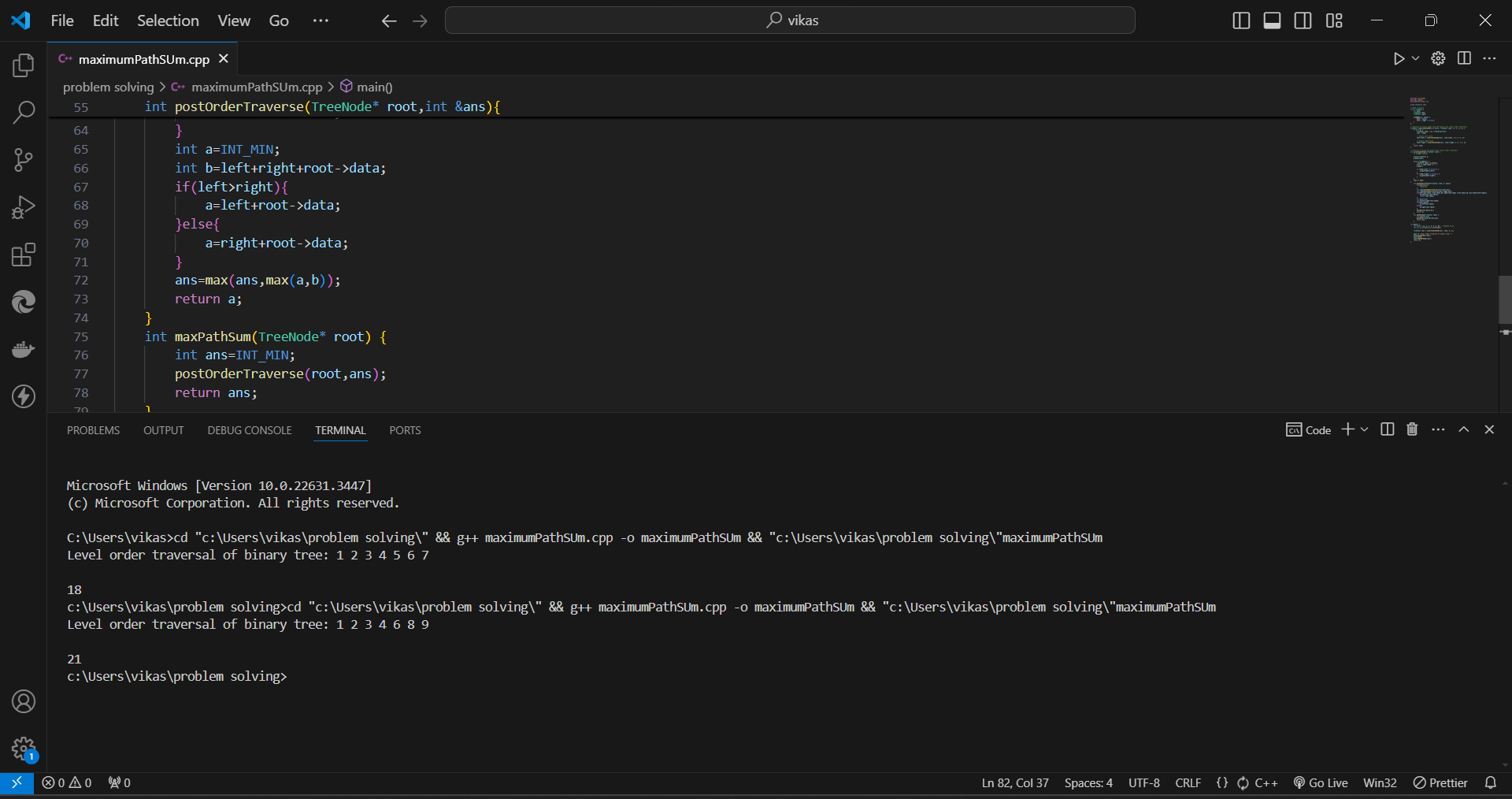
}

int maxPathSum(TreeNode\* root) {

int ans=INT\_MIN;

postOrderTraverse(root,ans);

return ans;

}  
  
Output::  
  


1. Design an algorithm to serialize and deserialize a binary tree. Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment. Implement the serialize and deserialize methods.  
     
     
   Serialize code ::  
     
   void serializeFunction(TreeNode\* root, ostringstream& out) {

if (!root) {

out << "# ";

return;

}

out <<root->data<<" ";

serializeFunction(root->left, out);

serializeFunction(root->right, out);

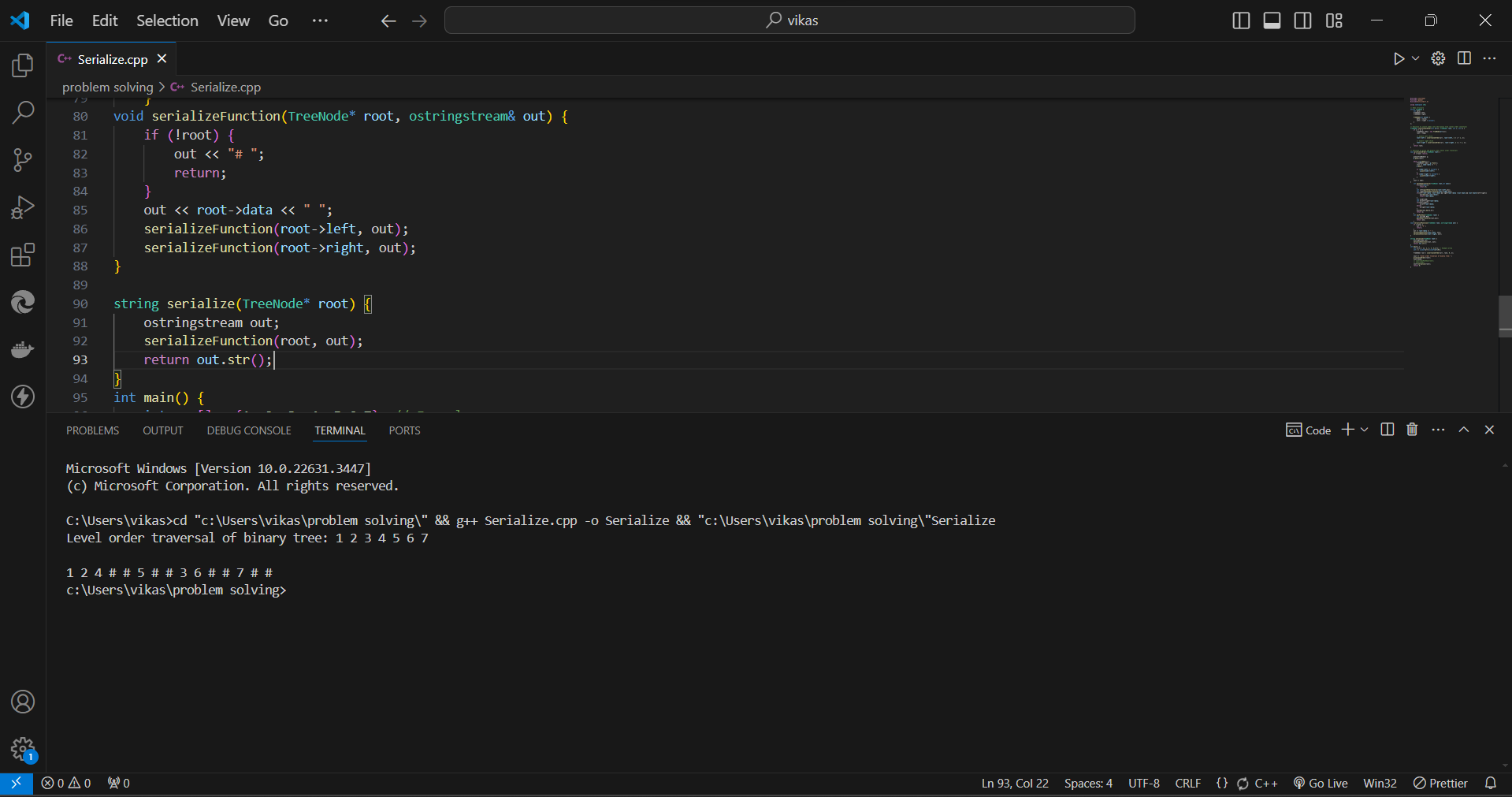
}

string serialize(TreeNode\* root) {

ostringstream out;

serializeFunction(root, out);

return out.str();

}  
  
OUTPUT::

Deserialize code  
  
TreeNode\* deserializerFunction(istringstream& in) {

string val;

in >> val;

if (val == "#") {

return nullptr;

}

TreeNode\* node = new TreeNode(stoi(val));

node->left = deserializerFunction(in);

node->right = deserializerFunction(in);

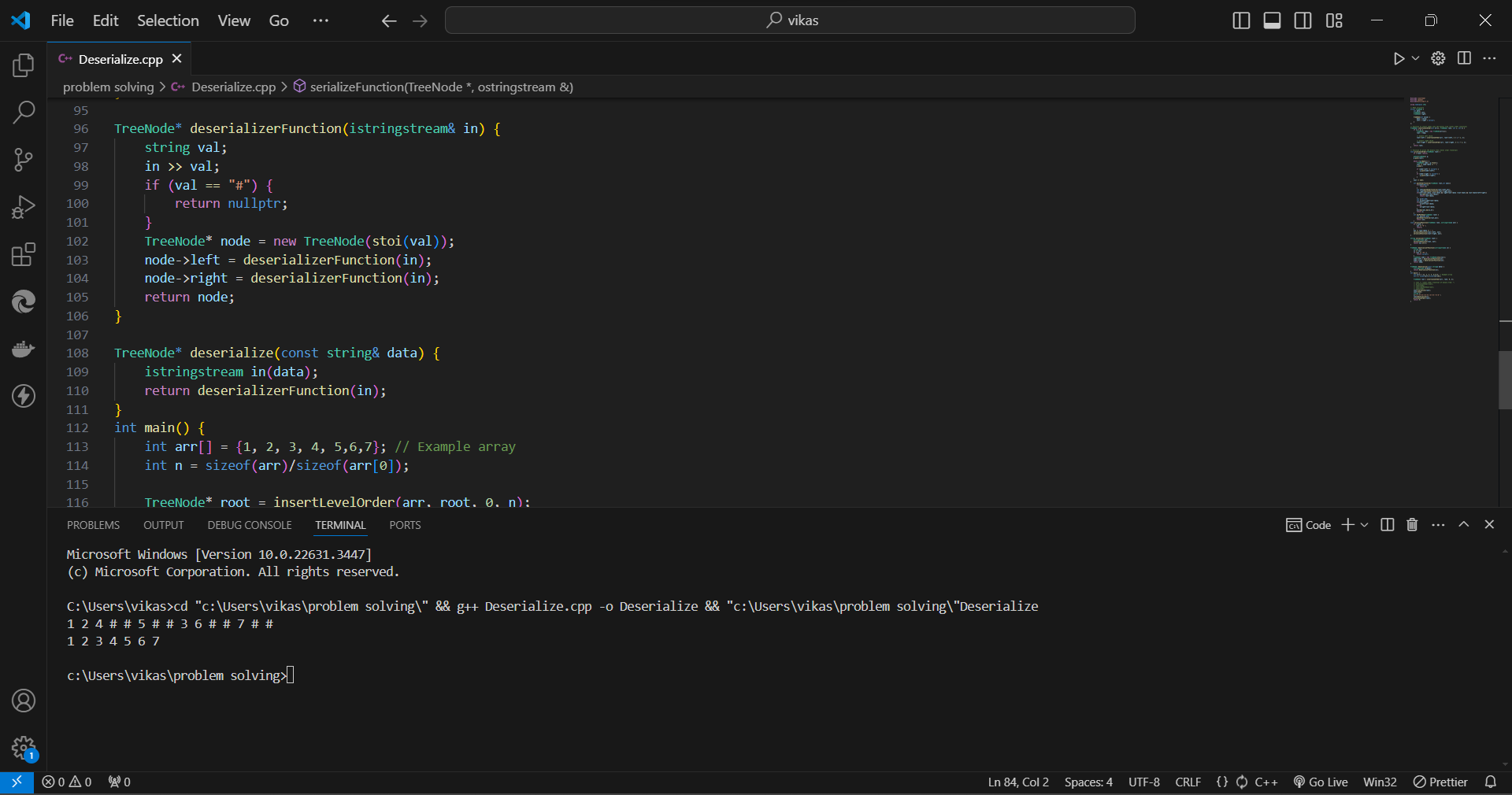
return node;

}

TreeNode\* deserialize(const string& data) {

istringstream in(data);

return deserializerFunction(in);

}  
  
  
Output::  


1. Write a function to rotate an array to the right by k steps.The function should modify the array in place to achieve the rotation.

Code::   
  
void rotateArray(vector<int>& nums, int k) {

int n=nums.size();

k = k%n;

int count = 0;

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

int current = i;

int prev = nums[i];

do {

int next = (current + k) % n;

int temp = nums[next];

nums[next] = prev;

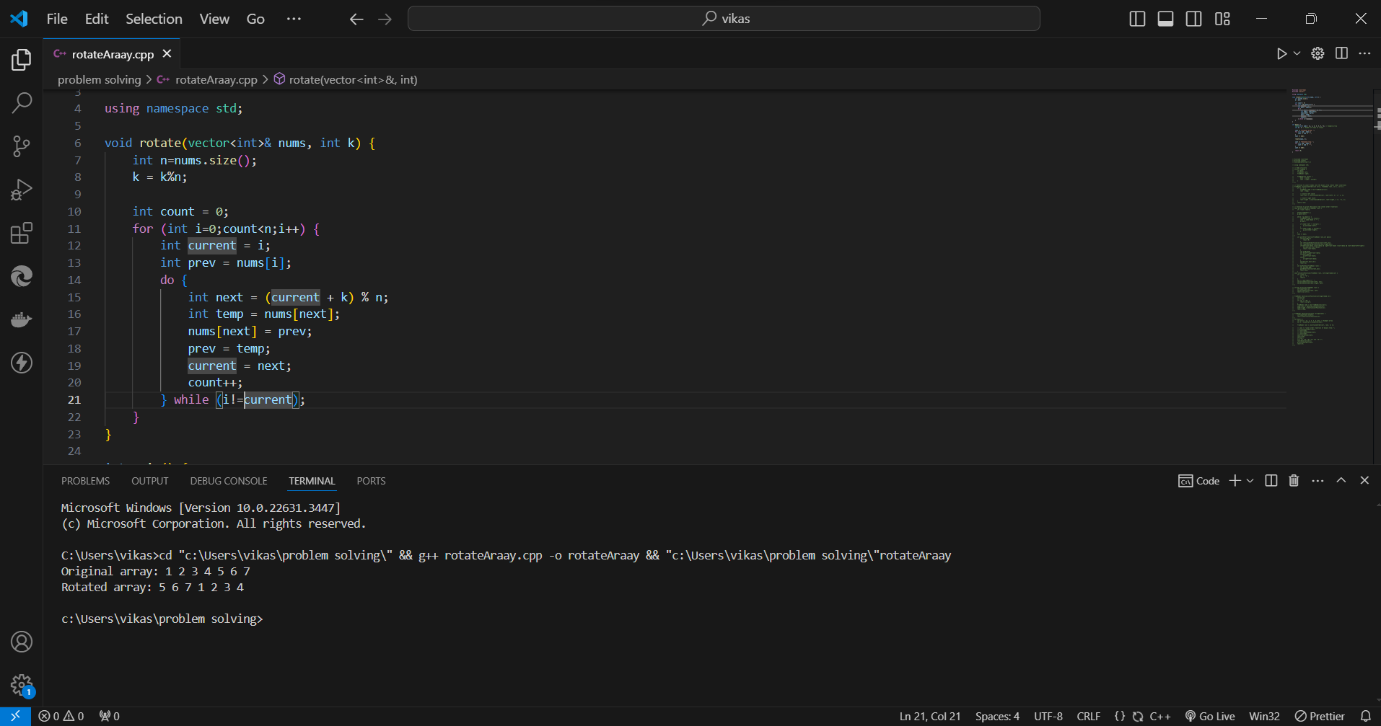
prev = temp;

current = next;

count++;

} while (i!=current);

}

}  
  


1. Write a function to find the factorial of a given number.The function should return the factorial of the number.  
     
   Code ::

int factorial(int n) {

if (n < 0) {

return -1;

}

if (n == 0 || n == 1) {

return 1;

}

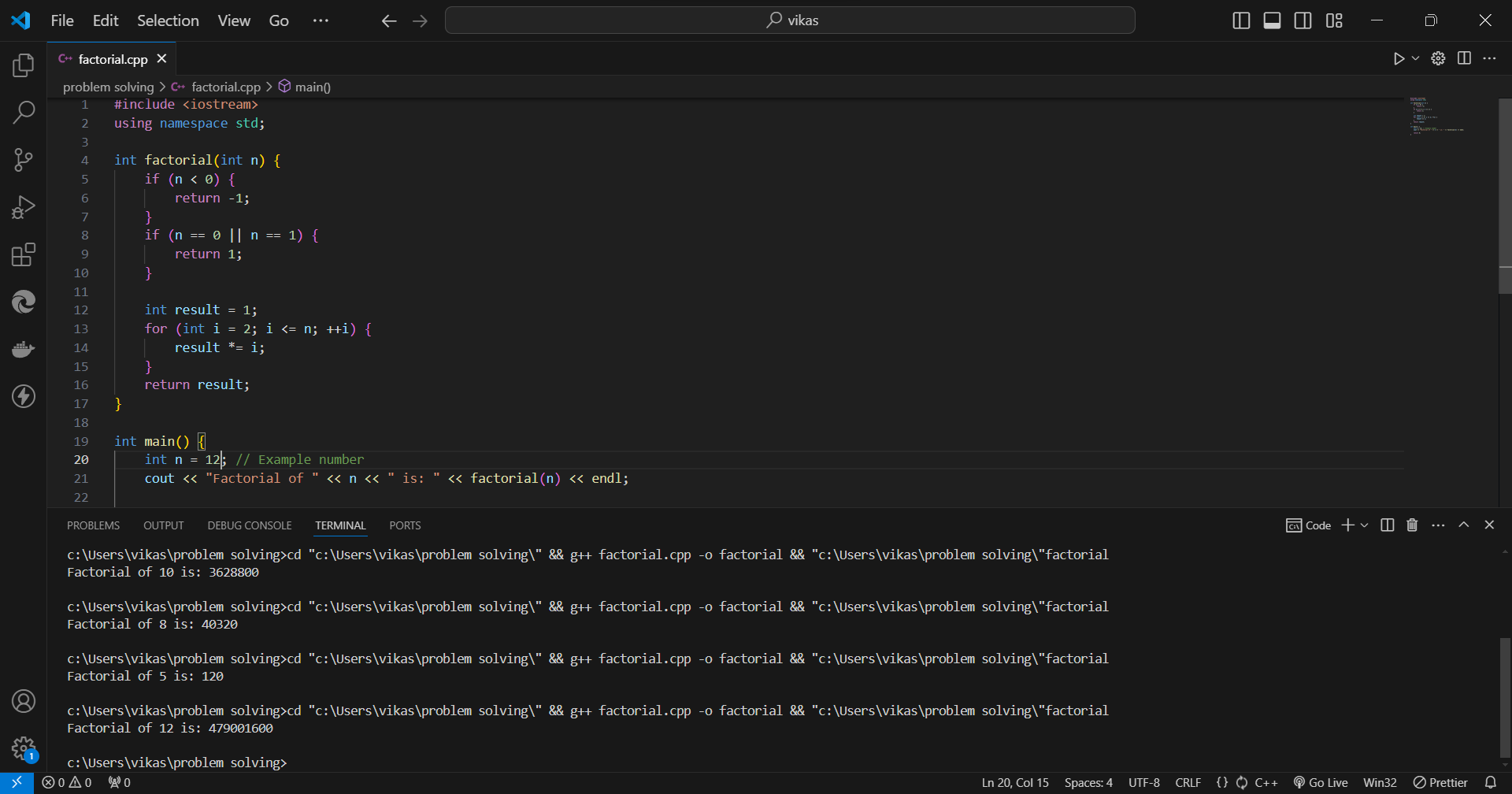
int result = 1;

for (int i = 2; i <= n; ++i) {

result \*= i;

}

return result;

}  
  
  
Output::   
  


1. Write a function to compute the sum of the digits of a given number.The function should return the sum of the digits of the number.

Code ::  
  
int sumOfDigits(int number) {

int sum = 0;

number = abs(number);

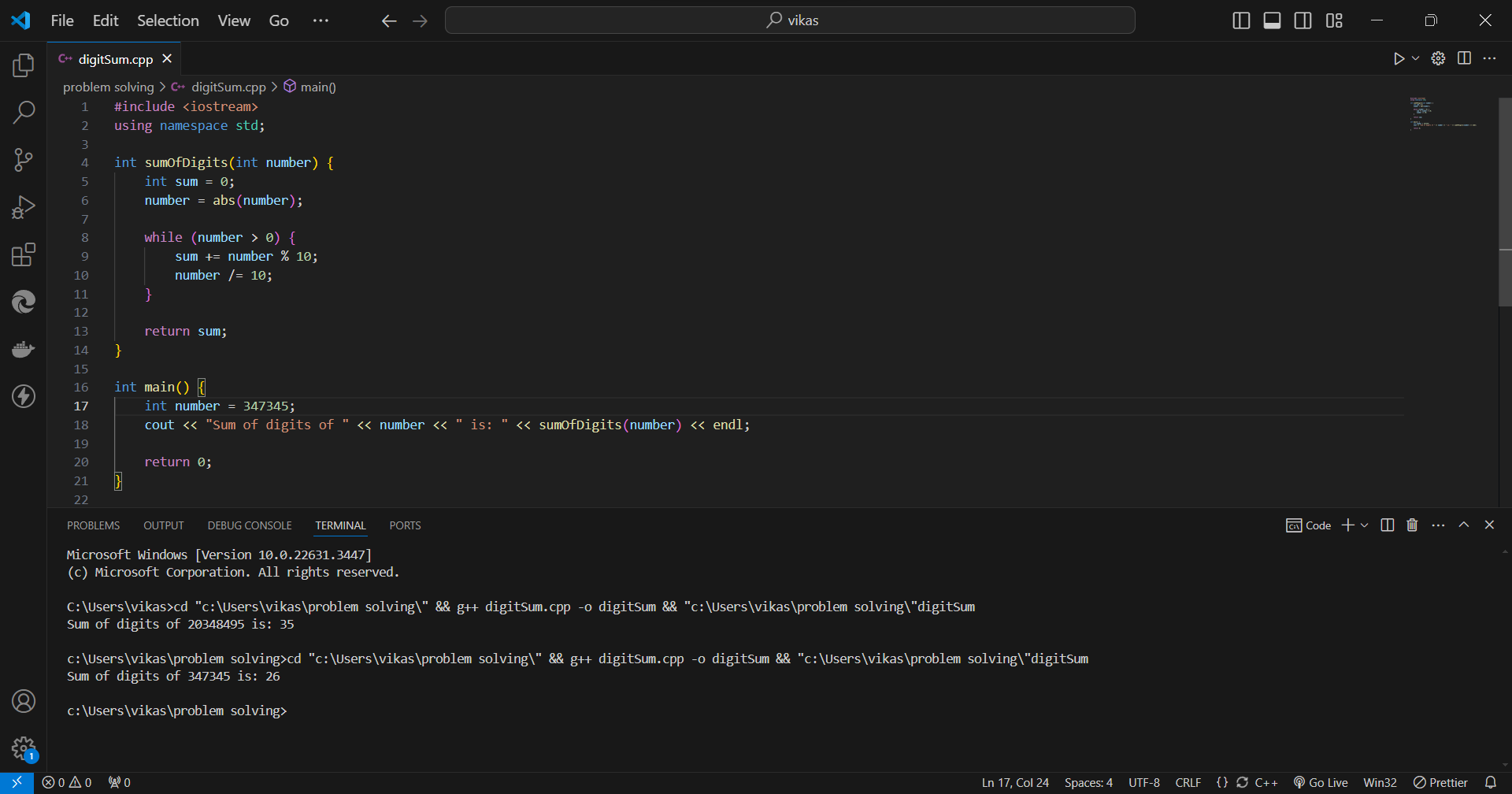
while (number > 0) {

sum += number % 10;

number /= 10;

}

return sum;

}  
  
Output ::  
  


1. Write a function to find the greatest common divisor (GCD) of two numbers. The function should return the GCD of a and b.

Code ::  
int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

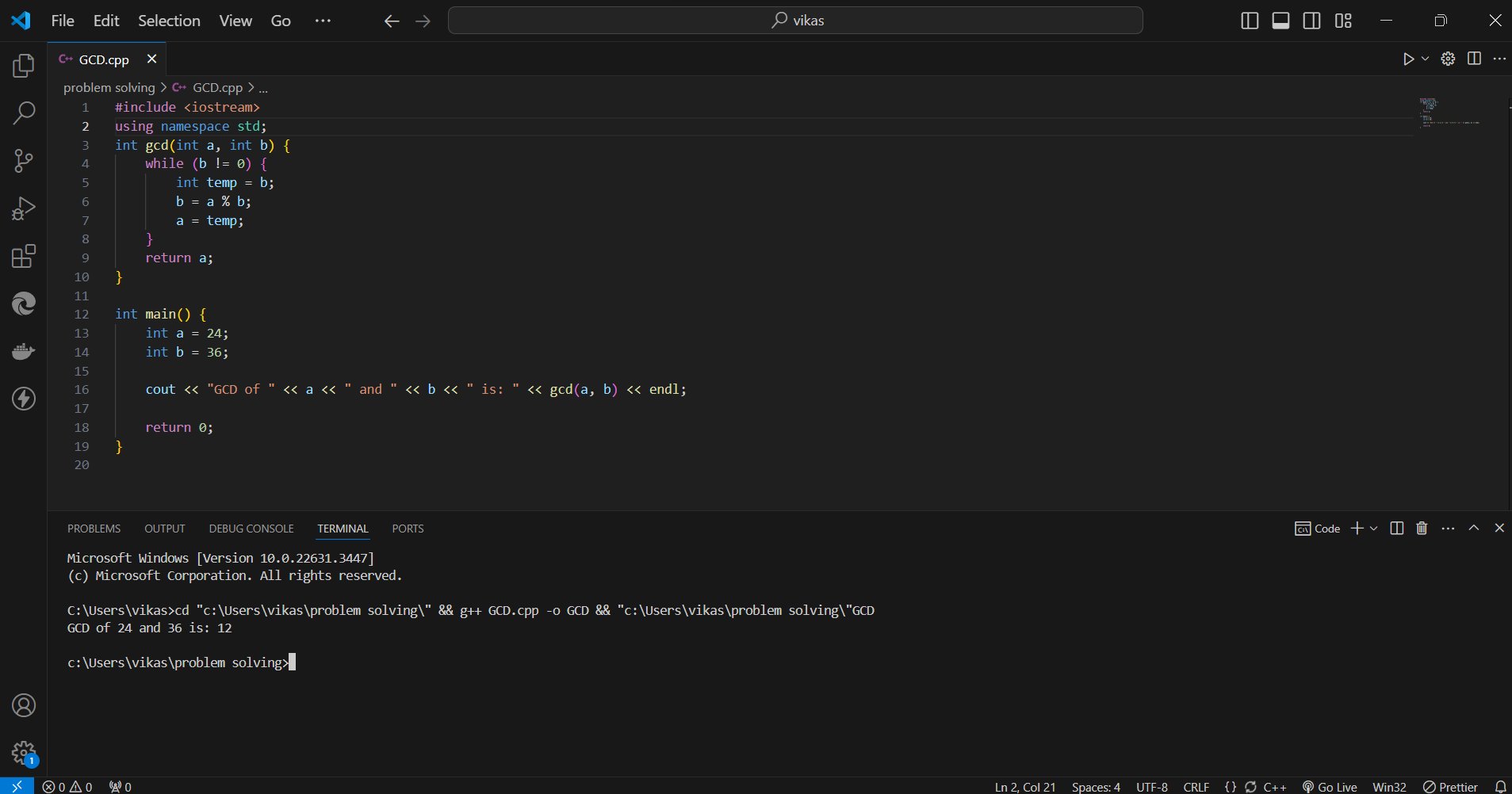
a = temp;

}

return a;

}

Output ::



1. Write a function to find the maximum difference between any two elements in an array.The function should return the maximum difference between any two elements in the array.

Code ::   
  
int maxDifference(const vector<int>& nums) {

int n = nums.size();

if (n < 2) {

return 0;

}

int min\_num = nums[0];

int max\_num = nums[0];

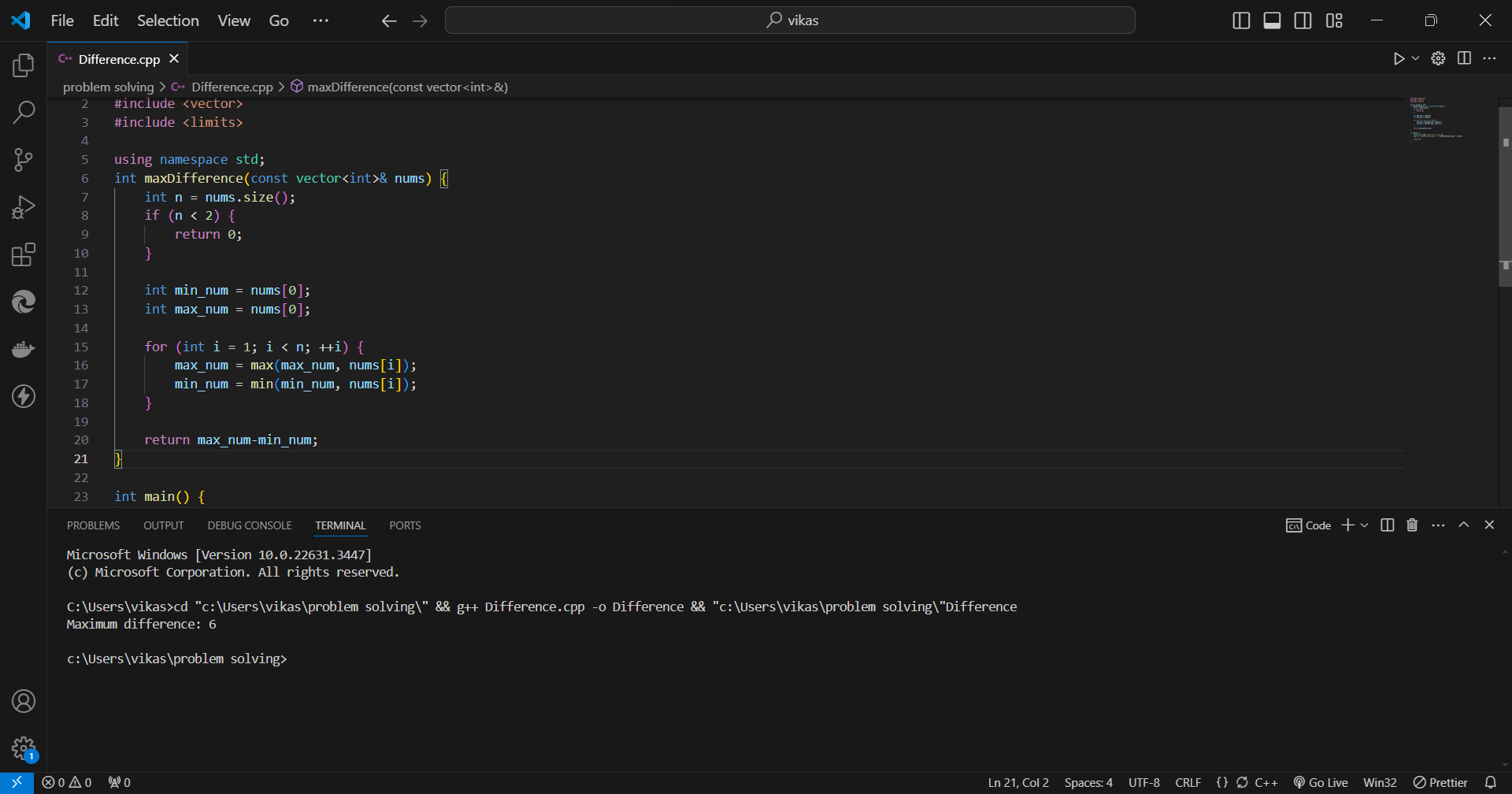
for (int i = 1; i < n; ++i) {

max\_num = max(max\_num, nums[i]);

min\_num = min(min\_num, nums[i]);

}

return max\_num-min\_num;

}  
  
  
Output ::  
  


1. Write a function to check if a given string contains only alphabetic characters.The function should return true if the string contains only alphabetic characters, and false otherwise.

Code ::  
bool containsOnlyAlphabetic(string s) {

int length=s.size();

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

if((s[i]<123 && s[i]>96) || (s[i]>64 && s[i]<91)){

continue;

}else{

return false;

}

}

return true;

}

Output ::  
