**ASSESSMENT**

**(Question/Answers and outputs)**

**Q1.** **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.**

**Ans1. Code:**

**#include<iostream>**

**using namespace std;**

**struct node{**

**int data;**

**node\* next;**

**node(int val):data(val),next(nullptr){}**

**};**

**node\* rev\_linkedlist(node\* head){**

**node\*previous=nullptr;**

**node\* current=head;**

**node\* next=nullptr;**

**while(current!=nullptr){**

**next=current->next;**

**current->next=previous;**

**previous=current;**

**current=next;**

**}**

**return previous;**

**}**

**void display(node\* head){**

**while(head!=nullptr){**

**cout<<head->data<<" ";**

**head=head->next;**

**}**

**cout<<endl;**

**}**

**int main(){**

**node\* head=new node(1);**

**head->next=new node(8);**

**head->next->next=new node(72);**

**head->next->next->next=new node(50);**

**cout<<"Orignal linkedlist:";**

**display(head);**

**head=rev\_linkedlist(head);**

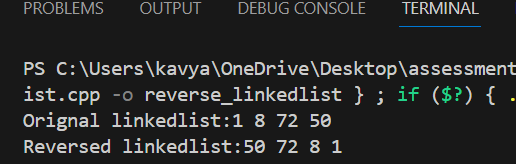
**cout<<"Reversed linkedlist:";**

**display(head);**

**return 0;**

**}**

**Output:**

****

**Q2.** **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.**

**Ans2. Code:**

**#include <iostream>**

**using namespace std;**

**class Solution {**

**public:**

**int length\_substr(string s) {**

**int max\_len=0;**

**int start=0;**

**int char\_index[256]={ -1 };**

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

**if (char\_index[s[end]]>=start) {**

**start=char\_index[s[end]]+1;**

**}**

**char\_index[s[end]]=end;**

**max\_len=max(max\_len,end-start+1);**

**}**

**return max\_len+1;**

**}**

**};**

**int main() {**

**Solution sol;**

**string str;**

**cout<<"Enter a string: ";**

**cin>>str;**

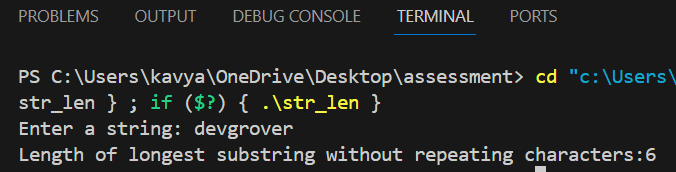
**int result=sol.length\_substr(str);**

**cout<<"Length of longest substring without repeating characters:"<<result<<endl;**

**return 0;**

**}**

**Output:**

****

**Q3.** **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.**

**Ans3. Code:**

**#include <iostream>**

**using namespace std;**

**struct TreeNode {**

**int val;**

**TreeNode \*left, \*right;**

**TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}**

**};**

**class Solution{**

**private:**

**int max\_sum=-2147483648;**

**public:**

**int maxPathSum(TreeNode\* root) {**

**cal\_pathsum(root);**

**return max\_sum;**

**}**

**int cal\_pathsum(TreeNode\* node) {**

**if (!node)**

**return 0;**

**int left\_max = max(0,cal\_pathsum(node->left));**

**int right\_max = max(0,cal\_pathsum(node->right));**

**int current\_max = node->val + (left\_max>right\_max?left\_max:right\_max);**

**int max\_through\_root = node->val+left\_max+right\_max;**

**max\_sum = max\_sum > max(max\_through\_root,current\_max)?max\_sum : max(max\_through\_root, current\_max);**

**return current\_max;**

**}**

**};**

**TreeNode\* newNode(int val) {**

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

**return node;**

**}**

**int main(){**

**TreeNode\* root = newNode(1);**

**root->left=newNode(2);**

**root->right=newNode(3);**

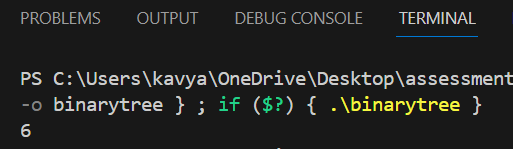
**Solution sol;**

**cout<<"max path sum:"<<sol.maxPathSum(root) << endl;**

**return 0;**

**}**

**Output:**

****

**Q4.** **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.**

**Ans4. Code:**

**#include <iostream>**

**#include <queue>**

**#include <string>**

**using namespace std;**

**struct TreeNode {**

**int value;**

**TreeNode\* left;**

**TreeNode\* right;**

**TreeNode(int val) : value(val), left(nullptr), right(nullptr) {}**

**};**

**class TreeSerializer {**

**public:**

**string encode(TreeNode\* root) {**

**string serialized;**

**encodeHelper(root, serialized);**

**return serialized;**

**}**

**TreeNode\* decode(string data) {**

**int index = 0;**

**return decodeHelper(data, index);**

**}**

**private:**

**void encodeHelper(TreeNode\* node, string& result) {**

**if (!node) {**

**result += "# ";**

**return;**

**}**

**result += to\_string(node->value) + " ";**

**encodeHelper(node->left, result);**

**encodeHelper(node->right, result);**

**}**

**TreeNode\* decodeHelper(string& data, int& index) {**

**if (data[index] == '#') {**

**index += 2;**

**return nullptr;**

**}**

**int start = index;**

**while (data[index] != ' ') {**

**index++;**

**}**

**int value = stoi(data.substr(start, index - start));**

**index++;**

**TreeNode\* node = new TreeNode(value);**

**node->left = decodeHelper(data, index);**

**node->right = decodeHelper(data, index);**

**return node;**

**}**

**};**

**void printTree(TreeNode\* root) {**

**if (!root) return;**

**cout << root->value << " ";**

**printTree(root->left);**

**printTree(root->right);**

**}**

**int main() {**

**TreeNode\* root = new TreeNode(1);**

**root->left = new TreeNode(2);**

**root->right = new TreeNode(3);**

**root->right->left = new TreeNode(4);**

**root->right->right = new TreeNode(5);**

**TreeSerializer codec;**

**string serialized = codec.encode(root);**

**cout << "Serialized tree: " << serialized << endl;**

**TreeNode\* deserialized = codec.decode(serialized);**

**cout << "Deserialized tree: ";**

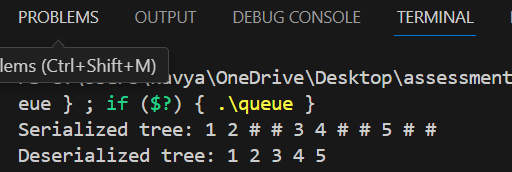
**printTree(deserialized);**

**cout << endl;**

**return 0;**

**}**

**Output:**

****

**Q5.** **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.**

**Ans5. Code:**

**#include <iostream>**

**using namespace std;**

**void rev\_array(int arr[],int start,int end) {**

**while (start < end) {**

**int temp = arr[start];**

**arr[start] = arr[end];**

**arr[end] = temp;**

**start++;**

**end--;**

**}**

**}**

**void rotate(int arr[],int n,int k) {**

**k = k % n;**

**if (k==0)**

**return;**

**rev\_array(arr,0,n-1);**

**rev\_array(arr,0,k-1);**

**rev\_array(arr,k,n-1);**

**}**

**int main(){**

**int n;**

**cout<<"enter array size:";**

**cin>>n;**

**int arr[n];**

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

**cout<<"Enter array "<<i+1<<" element:";**

**cin>>arr[i];**

**}**

**int k;**

**cout<<"Enter the value of k(steps):";**

**cin>>k;**

**rotate(arr,n,k);**

**cout<<"Rotated array:";**

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

**cout << arr[i] << " ";**

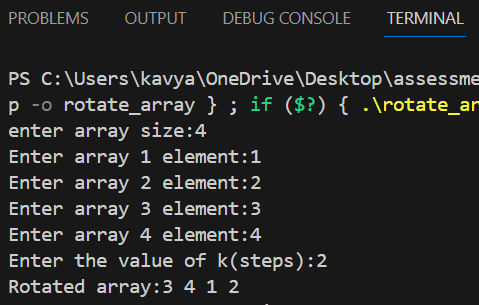
**}**

**cout<<endl;**

**return 0;**

**}**

**Output:**

****

**Q6.** **Write a function to find the factorial of a given number. The function should return the factorial of the number.**

**Ans6. code:**

**#include<iostream>**

**using namespace std;**

**class factorial{**

**public:**

**long long fact=1;**

**int i;**

**long long cal\_factorial(int num){**

**if(num<0){**

**cout<<"Number cannot be negative";**

**}**

**for(i=1;i<=num;i++)**

**{**

**fact\*=i;**

**}**

**return fact;**

**}**

**};**

**int main(){**

**factorial f;**

**int num;**

**cout<<"Enter a number:";**

**cin>>num;**

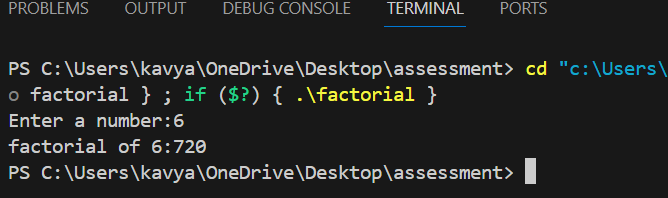
**long long result=f.cal\_factorial(num);**

**cout<<"factorial of "<<num<<":"<<result;**

**return 0;**

**}**

**Output:**

****

**Q7. 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.**

**Ans7. Code:**

**#include<iostream>**

**using namespace std;**

**class sum\_of\_digit{**

**public:**

**int remainder;**

**int sum=0;**

**int i;**

**int cal\_sum(int num)**

**{**

**while(num!=0)**

**{**

**remainder=num%10;**

**sum+=remainder;**

**num=num/10;**

**}**

**return sum;**

**}**

**};**

**int main()**

**{**

**sum\_of\_digit s;**

**int num;**

**cout<<"Enter a number:";**

**cin>>num;**

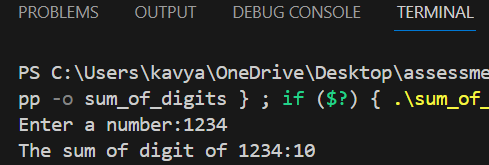
**int result=s.cal\_sum(num);**

**cout<<"The sum of digit of "<<num<<":"<<result;**

**return 0;**

**}**

**Output:**

****

**Q8.** **Write a function to find the greatest common divisor (GCD) of two numbers. The function should return the GCD of a and b.**

**Ans8. Code:**

**#include<iostream>**

**using namespace std;**

**class gcd\_no{**

**public:**

**int temp;**

**int cal\_gcd(int a,int b){**

**if(a<b){**

**temp=a;**

**a=b;**

**b=temp;**

**}**

**if(b==0){**

**return a;**

**}**

**return cal\_gcd(b,a%b);**

**}**

**};**

**int main(){**

**gcd\_no g;**

**int a,b;**

**cout<<"Enter 1st number:";**

**cin>>a;**

**cout<<"Enter 2nd number:";**

**cin>>b;**

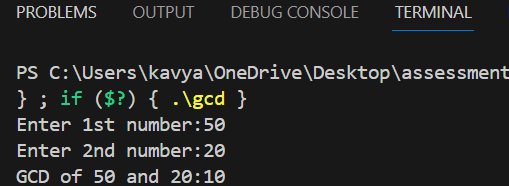
**int result=g.cal\_gcd(a,b);**

**cout<<"GCD of "<<a<<" and "<<b<<":"<<result;**

**return 0;**

**}**

**Output:**

****

**Q9.** **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.**

**Ans9. Code:**

**#include<iostream>**

**using namespace std;**

**class max\_diff{**

**public:**

**int max,min;**

**int diff(int a[],int size){**

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

**for(int j=0;j<size-i-1;j++){**

**if(a[j]>a[j+1]){**

**int temp=a[j];**

**a[j]=a[j+1];**

**a[j+1]=temp;**

**}**

**}**

**min=a[0];**

**max=a[size-1];**

**return (max-min);**

**}**

**}**

**};**

**int main(){**

**max\_diff m;**

**int n;**

**cout<<"Enter array size:";**

**cin>>n;**

**if(n<2){**

**cout<<"NOt Allowed:Array must contain atleast 2 elements.";**

**}**

**int a[n];**

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

**cout<<"Enter array "<<i+1<<" element:";**

**cin>>a[i];**

**}**

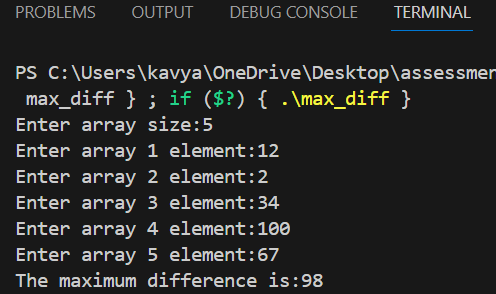
**int result=m.diff(a,n);**

**cout<<"The maximum difference is:"<<result;**

**return 0;**

**}**

**Output:**

****

**Q10.** **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.**

**Ans10.** **#include<iostream>**

**#include<string>**

**using namespace std;**

**class string\_check{**

**public:**

**bool check\_char(string &str){**

**for(char c:str){**

**if(!isalpha(c)){**

**return false;**

**}**

**}**

**return true;**

**}**

**};**

**int main(){**

**string\_check s;**

**string str;**

**cout<<"Enter a string:";**

**cin>>str;**

**if(s.check\_char(str)){**

**cout<<"String contains only Alphabet characters.";**

**}**

**else{**

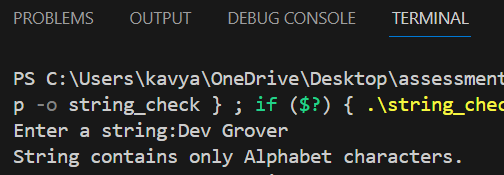
**cout<<"String does not contains only Alphabet characters.";**

**}**

**return 0;**

**}**

**Output:**

****