**Debugging Questions**

**Question: 1**

This is a function that checks if the entered number is a palindrome or not, a palindrome number is a number which is the same when its digits are reversed

eg) 1234321

**Java:**

**Wrong code:**

void palindrome(n)

{

int rev = 0, temp = 0; while(n>=0)

{

rev = rev%10+n\*10; n -= 10;

}

if(rev == temp) System.out.println("palindrome");

else

System.out.println("not palindrome");

}

**Input:**

12321

**Output:** palindrome

**Question: 2**

## **Python has a module named "re" to work with RegEx(Regular Expressions) In this Code We Are Using RegEx To Find The Type Of IP Address It Is. The Code Below Checks Whether It Is IPV4 Or IPV6 Address.**

**WRONG CODE TO BE DEBUGGED:**

*#WRONG CODE*

*# re module provides support for regular expressions*

import re

*# Regular expression for validating an Ipv4*

ipv4 = "^(25[0-5]|25[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)$"

*# Regular expression for validating an Ipv6*

ipv6 = "(([0-9a-fA-F]{1,4}:){1,7}[0-9a-fA-F]{1,4}|

([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:)

{1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1

,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}

:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{

1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA

-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a

-fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6})|:((:[0

-9a-fA-F]{1,4}){1,7}|:)|fe80:(:[0-9a-fA-F]{0,

4}){0,4}%[0-9a-zA-Z]{1,}|::(ffff(:0{1,4}){0,1}

:){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9

])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0

-9])|([0-9a-fA-F]{1,4}:){1,4}:((25[0-5]|(2[0-4]

|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]

|1{0,1}[0-9]){0,1}[0-9]))"

**def** ip\_type(Ip):

**if** re.search(ipv4, Ip): print("IPv4")

**elif** re.search(ipv6, Ip): print("IPv6")

### **else:**

print("Neither")

*# Driver Code*

**if** name == ' main ' :

*# Enter the Ip address*

Ip = "192.0.2.126"

*# calling run function*

ip\_type(Ip)

Ip = "3001:0da8:75a3:0000:0000:8a2e:0370:7334"

ip\_type(Ip)

Ip = "36.12.08.20.52"

ip\_type(Ip)

**Input:**

None

**Output:**

IPv4

IPv6

Neither

# 

# 

# 

# 

# 

# 

# **Question: 3**

## **lovely\_list() is a function to sort a provided list in ascending order.**

## 

## **Wrong code:**

*#WRONG CODE*

**def** lovely\_list(lis): n = len(lis)

**for** x **in** range(n):

flag = False

**for** j **in** range(0, x-1):

**if** lis[j] > lis[j+1] :

lis[j], lis[j+1] = lis[j+1], lis[j] flag = True

**if** flag == False:

### **continue**

tes\_lis = [5, 28, 42, 75, 114, 16, 80]

lovely\_list(tes\_lis) print ("Sorted list :")

**for** i **in** range(len(tes\_lis)):

print ("%d" %tes\_lis[i],end=" ")

**Input:**None

**Output:**Sorted List :   
11 12 22 25 34 64 90

**Question: 4**

Pattern

If the value of n is 5

1

212

32123

4321234

543212345

**Wrong code:**

n=int(input()) for z in range(1,n+1):

for j in range(1,n-z+1): print(" ")

for j in range(z,0,-1): print(j,end="")

for j in range(2,z+1): print(j,end="")

print()

**INPUT:** 4

**OUTPUT:**

1

212

32123

4321234

**Question: 5**

Given a 2D integer array of size M\*N, find and print the sum of ith column elements separated by space.

Input Format :

First and only line of input contains M and N, followed by M \* N space separated integers representing the elements in the 2D array.

Output Format :

Sum of every ith column elements (separated by space)

Constraints :

1 <= M, N <= 10^3

**Sample Input :**

4 2 1 2 3 4 5 6 7 8

**Sample Output :**

16 20

**Wrong code:**

#include<iostream> using namespace std;

int main()

{

int b[10][10]

int m,n

cin>>m>>n

for(int i=0;i<=m){

for(int j=0;j<=n){

cin>>b[j][i]

}

}

for(int d=1;d<=n){

for(int e=1;e<=m){

sum+=b[d][e]

}

cout<<sum

}

}

**Question: 6**

Two integers n are k are taken via input

Check whether the kth bit in the binary representation of n is set(turned on / has a value of 1) or not

If the kth bit is turned on then print "the bit is set" otherwise print "the bit is unset” (here, the indexing of the binary representation of the number(n) starts form 1) (CPP)

**Sample Input:**

45

2

**Sample Output:**

the bit is set

**Wrong code:**

#include <stdio.h>

#include <iostream>

using namespace std;

int main()

{

int n,k,c;

cout<<"Enter the number"<<endl; cin>>n;

cout<<"Enter the value of k"<<endl; cin>>k;

c=n&(1>>(k)); if ( c==0){

cout<<"the bit is set "<<endl;

}else{

cout<<"the bit is unset"<<endl;

}

return 0;

}

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **Question: 7**

This is a program that searches an element in the list using the binary search method. A binary search method is a method in which a sorted list is taken and the element to be found is compared with the middle element in the list. If the element is greater than the middle element, the first half of the list is removed, and if the element is smaller than the middle element, then the second half of the list is removed. This process is repeated until the element is found or not found.

**Sample input:**

4

1

2

3

4

3

**Sample output:**

[1, 2, 3, 4]

Element is present at index 2

### **Python:**

**Wrong code:**

def binarySearch(nums, min, max, elem): mid = (max + min) % 2

if(min < max):

print("Element is not present in array") elif(elem < nums[mid]):

binarySearch(nums, mid - 1, max, elem) elif(elem > nums[mid]):

binarySearch(nums, min, mid + 1, elem) else:

print(f"Element is present at index {min}")

n = int(input()) nums = []

for i in range(n): nums.append(int(input()))

nums.sort()

print("Initially array is:", nums)

elem = int(input("element to be searched:")) print(elem)

binarySearch(nums, 0, len(nums)-1, elem)

**Question: 8**

Reverse the given string word wise. That is, the last word in given string should come at 1st place, last second word at 2nd place and so on. Individual words should remain as it is.

Input format :

String in a single line Output format :

Word wise reversed string in a single line Constraints :

0 <= |S| <= 10^7

where |S| represents the length of string, S

**Sample Input 1:**

Welcome to GFG BU

**Sample Output 1:**

BU GFG to Welcome

**Wrong code**

#include <iostream>

using namespace std;

void reverseStringWordWise(char input[])

{

int temp;

int len=0;

int i=0;

while (input[i] != '\0'){

len++;

}

int start = 0, end = len - 1;

while (start<=end){

temp= input[start];

input[start]=input[end];

input[end]=temp;

start - -;

end- -;

}

i=0;

start=0;

while(input[i] != '\0'){

if(input[i]==' '){

end=i-1;

while(start<end){

temp= input[end];

input[end]=input[start];

input[start]=temp;

start - -;

end++;

}

start=end-1;

}

}

word

end=len;

while(start<=end){

temp= input[end];

input[start]=input[end];

input[start]=temp;

start - -;

end - -;

}

}

int main() {

char input[1000];

cin.getline(input, 1000);

reverseStringWordWise(input);

cout << input << endl;

}

**Question: 9**

The following code is written to print all the possible subsets of the number ‘num’.

**Input :**

123

**Output :**

{}

{1}

{2}

{3}

{12}

{13}

{23}

{123}

### **Python:**

**Wrong code:**

def func(sen, k, t):

if(k == 0):

print("{t}") return

for i in range(0, len(sen)+1): func(sen[:i+1], t+sen[i], k)

num = 123

sen = str(num)+""

for i in range(0, len(sen)+1): func(sen, "", i)