CodeQuotient

Department of Computer Science & Engineering

LAB MANUAL

22CS006-Object Oriented Programming in C - 3rdSem

Student Name	**ANUJ KUMAR**
Email	**anuj1699.be23@chitkara.edu.in**
Course Name	**22CS006-Object Oriented Programming in C - 3rdSem**

Faculty Incharge

_____

Table of Contents

Aim: Factorial of a number

Given a number **n**, compute n factorial (written as n!) where n >=0.
A factorial of a number n is the product of all natural numbers from 1 to n.
For Example , factorial of 4 (4!) = 1*2*3*4 = 24
Input Format:
Each test case will contains an integer n where n >=1.
Constraints
0 <= n <= 15
Output Format:
For each input case, calculate and print the factorial of n.
Sample Input
4
Sample Output
24

Solution:

```
long fact(int n) {
   if (n == 0) {
      return 1;
   }
   else {
      return n * fact(n - 1);
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Sum of a set of numbers

Given a set of n numbers , write a program to add these numbers and return the resultant sum.

Input Format:

First line of input of each test case will contain a number N = number of elements in the set.
Next N lines will contain number n where n>=0.

Output Format:

For each input case, sum and print the all the numbers.

Sample Input

5
1
2
3
4
5

Sample Output

15

Solution:

```
#include <iostream>
using namespace std;
int main() {
   int N;
   cin >> N;
   int sum = 0;
   int num;
   for (int i = 0; i < N; ++i) {
      cin >> num;
      sum += num;
   }
   cout << sum << endl;
   return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Rotate a list

Given an array of integers and a positive integer R, rotate the array by shifting each element by R positions towards the left in a circular manner.

Input

First line will contain the number of test cases. Each test case will contain 3 lines. First line of the test case will contain a number N = number of elements in the array (1 <= N <= 50). Next line will contain N space separated integers. The third line will contain an integer R = number of shifts in rotation. It is possible that R is greater than N.

Output

Print one line of output for each test case with the integers in the rotated array printed as space separated list.

There should be no space after the last element.

Sample Input:

```
3
7
1 2 3 4 5 6 7
3
9
3 5 2 1 7 5 9 15 11
10
1
5
13
```

Sample Output

```
4 5 6 7 1 2 3
5 2 1 7 5 9 15 11 3
5
```

Solution:

```
#include <iostream>
#include <vector>
using namespace std;
void rotateArray(vector<int>& arr, int N, int R) {
    int rotations = R % N;
    vector<int> rotated(N);
    for (int i = 0; i < N; ++i) {
        rotated[(i + N - rotations) % N] = arr[i];
    }
    for (int i = 0; i < N; ++i) {
        cout << rotated[i];
        if (i < N - 1) {
            cout << " ";
        }
    }
    cout << endl;
}
int main() {
    int T;
    cin >> T;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    while (T--) {
        int N;
        cin >> N;
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) {
            cin >> arr[i];
        }
        int R;
        cin >> R;
        rotateArray(arr, N, R);
    }
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Maximum element in an Array

Write a program to find the maximum element in an array.
Input Format:
The first line of input contains an integer N, denoting the size of array.
The second line contains N space separated integers, denoting the array elements.
Output Format:
For each test case, print the maximum element in the array.
Constraints
1 <= N <= 10^5
-1000 <= arr[i] <= 1000
Sample Input 1
7
20 30 93 71 18 82 66
Sample Output 1
93
Sample Input 2
5
33 45 67 33 67
Sample Output 2
67

Solution:

```
int maxElement(int arr[], int N) {
    int maxElem = arr[0];
    for (int i = 1; i < N; ++i) {
        if (arr[i] > maxElem) {
            maxElem = arr[i];
        }
    }
    return maxElem;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Kth largest number

Given a method called **kthLargest** that accepts an integer ***k*** and an array ***arr*** as its parameters and returns the element such that *k* elements have greater or equal value. If k = 0, return the largest element; if k = 1, return the second largest element, and so on.
For example, if the array passed contains the values {74, 85, 102, 99, 101, 56, 84} and the integer *k* passed is 2, your method should return 99 because there are two values at least as large as 99 (101 and 102).
Expected Time Complexity: O(N)
Assume that 0 <= k < length of array.
Input Format
First line contains the number of test cases i.e. T
Each test case T contains the integer K (In first line), the number of elements in array N (In 2nd line) and N array integers in the next line.
Output Format
Print the kth largest number from array.
Constraints
0 <= K < N
1 <= N <= 10^5
-(10^9) <= arr[i] <= 10^9
Sample Input
1 // Test cases
1 // K
4 // Size of Array
8 5 6 3 // Elements of Array
Sample Output
6

Solution:

```
int kthLargest(int arr[], int size, int k) {
    int left = 0, right = size - 1;
    while (left <= right) {
        int pivot = arr[right];
        int i = left - 1;
        for (int j = left; j < right; j++) {
            if (arr[j] >= pivot) {
                i++;
                swap(arr[i], arr[j]);
            }
        }
        swap(arr[i + 1], arr[right]);
        int pivotIndex = i + 1;
        if (pivotIndex == k) {
            return arr[pivotIndex];
        } else if (pivotIndex < k) {
            left = pivotIndex + 1;
        } else {
            right = pivotIndex - 1;
        }
    }
    return -1;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Matrix Multiplication

Given two matrices **A** and **B** in the form of two dimensional arrays, find the dot product of these two matrices.
[image:https://lh6.googleusercontent.com/ZniX6vnqQvQlnST1H8azWXBOJoFHVjl86AzIhXdSEkQG1jMamL4q_kHAw9fUojGXphn0NE0WD_Jna13_HdznlF06EiP3WG8YYEZuRWhQmv_O2OODz91D22_muARFcgcY_Cv_ZMAOXovDK_gEXfn5Tj4RPU3IhS1m3fiICXM6CT5apGhSarGXVQjwXWdo]

Input Format:
First line of input will contain an integer T = no. of test cases.
Each test case will follow in multiple lines. First line of each test case will contain two integers R1 = no. of rows in matrix A and C1 = no. of columns in matrix A. Next R1 lines will each contain C1 space separated integers. Next line again contain two integers R2 = no. of rows in B and C2 = no. columns in B. C1 will be equal to R2.
Output Format:
For each test case print matrix of size R1*C2 in R1 lines with each line containing C2 space separated integers of corresponding row.
Constraints
1 <= T <= 1000
1 <= R1, C1, R2, C2 <= 50
C1 will be equals to R2
Sample Input
2 // Test Cases
2 3 // R1 C1 (testcase 1)
1 2 3 // matrix A
4 5 6
3 2 // R2 C2
2 3 // matrix B
1 2
2 1
2 2 // R1 C1 (testcase 2)
12 4 // matrix A
7 6
2 3 // R2 C2
2 4 6 // matrix B
3 5 7
Sample Output
10 10 // A * B (testcase 1)
25 28
36 68 100 // A * B (testcase 2)
32 58 84

Solution:

```
void multiplyMatrix(int A[50][50], int B[50][50], int R1, int C1, int R2, int C2)
{
   int result[50][50] = {0};  // Initialize result matrix with zeros
   for (int i = 0; i < R1; ++i) {
      for (int j = 0; j < C2; ++j) {
         for (int k = 0; k < C1; ++k) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    for (int i = 0; i < R1; ++i) {
        for (int j = 0; j < C2; ++j) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Spirally traversing a matrix

Traversing an array is an elementary operation on an array, in which each element will be processed for some operation. Printing elements is one example operation.
Complete the function **printSpiral()** given in the editor, which accepts a two dimensional array and prints the array in spiral form rotating clockwise.

Input Format

Each test-case will begin with two number m and n where m = no. of rows and n = no. of columns.
m rows will follow with n integers in each row separated by a space.

Constraints

1<=m<=50 and 1<=n<=50.

Output Format

For each test case, print the elements of 2-d array in spiral form starting from index (0,0) or upper-left corner in clockwise direction.

Sample Input 1

```
3 3
4 5 6
7 8 9
10 11 12
```

Sample Output 1

```
4
5
6
9
12
11
10
7
8
```

Sample Input 2

```
3 4
4 3 2 1
5 7 8 11
2 4 6 8
```

Sample Output 2

```
4
3
2
1
11
8
6
4
2
5
7
8
```


Solution:

ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
void printSpiral(int a[50][50], int r, int c) {
    int top = 0, bottom = r - 1;
    int left = 0, right = c - 1;
    while (top <= bottom && left <= right) {
        // Traverse from left to right along the top row
        for (int i = left; i <= right; ++i) {
            cout << a[top][i] << endl;
        }
        top++;
        // Traverse from top to bottom along the right column
        for (int i = top; i <= bottom; ++i) {
            cout << a[i][right] << endl;
        }
        right--;
        // Traverse from right to left along the bottom row
        if (top <= bottom) {
            for (int i = right; i >= left; --i) {
                cout << a[bottom][i] << endl;
            }
            bottom--;
        }
        // Traverse from bottom to top along the left column
        if (left <= right) {
            for (int i = bottom; i >= top; --i) {
                cout << a[i][left] << endl;
            }
            left++;
        }
    }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find occurrences of palindrome words in a string

Write a function which given a string, count and return the palindrome words present in the string. A word in a string is separated with space(s).

Sample Input 1

Mom and Dad are my best friends

Sample Output 1

2

Explanation 1

þÿThis string contains two palindrome words (i.e., Mom, Dad) so the count is 2.

Sample Input 2

mohit speaks english

Sample Output 2

0

Explanation 2

This string contains no palindrome words.

Solution:

```
#include<sstream>
bool isPalindrome(const string &word) {
    int left = 0;
    int right = word.size() - 1;
    while (left < right) {
        if (tolower(word[left]) != tolower(word[right])) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
int countPalindrome(string str) {
    stringstream ss(str);
    string word;
    int count = 0;
    while (ss >> word) {
        if (isPalindrome(word)) {
            count++;
        }
    }
    return count;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Capitalize the first letter of each word

Write a program which given a string, Capitalize first letter of each word in it.
Sample Input
code quotient
Sample Output
Code Quotient
Sample Input
get better at coding
Sample Output
Get Better At Coding

Solution:

```
#include <string>
void capitalizeFirstChar(string &str) {
   bool capitalize = true;
   for (int i = 0; i < str.length(); ++i) {
      if (isspace(str[i])) {
         capitalize = true;
      } else if (capitalize && isalpha(str[i])) {
         str[i] = toupper(str[i]);
         capitalize = false;
      }
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Reverse a string

Given a string, the task is to reverse it.
For example, if the string is **"Hello"** , reversed string is **"olleH".**
Complete the function **reverseString()** given in the editor that accepts a string & reverses it.
Input Format:
First line inputs number of Testcases t
Then t strings follow in each line
Output Format:
For each test case output the reversed string
Sample Input 1
1
codequotient
Sample Output 1
tneitouqedoc
Sample Input 2
1
programming
Sample Output 2
gnimmargorp

Solution:

```
#include <string>
void reverseString(string &str) {
   int left = 0;
   int right = str.length() - 1;
   while (left < right) {
      swap(str[left], str[right]);
      left++;
      right--;
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Sum of Range

Given 2 numbers min & max, the task is to find the sum of the elements present in the range [min,max].
Complete the function **sumOfRange()** that accepts two integer parameters *min* and *max* and returns the sum of the integers from *min* through *max* inclusive.
For example, the call of **sumOfRange(3, 7)** returns 25 (3 + 4 + 5 + 6 + 7) . If *min* is greater than *max*, return 0.

Input Format

The first line of input contains the number of testcases, T
Then T lines follow, each line contains 2 integers, min & max, separated by a space

Output Format

For each testcase, print the sum of elements present in the range [min,max]

Sample Input

2
3 7
0 5

Sample Output

25
15

þÿExplanation:

For the first case , the sum of range [3,7] is (3+4+5+6+7) = 25
For the second case , the sum of range [0,5] is (0+1+2+3+4+5) = 15

Solution:

```
int sumOfRange(int min, int max) {
   if (min > max) {
      return 0;
   }
   int n = max - min + 1;
   int sum = n * (min + max) / 2;
   return sum;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Verify Prime Number

Given an integer n, the task is to check whether that the given integer is a prime number or not
Complete the function **verifyPrime()** given in the editor, that accepts an integer parameter n and returns true/1 if number is prime and false/0 if number is not prime.

Your solution must be optimized otherwise you may get an error "Time Limit Exceeded".
þÿ
Input Format:
The first line of input contains the number of test cases , T
Then T lines follow, each line contains an integer, n which is to be checked whether it is prime or not
Constraints:
1 "d T "d 100
0 "d n "d 10^9
Output Format:
Print 'PRIME', if the number is a prime number else 'NOT PRIME' for each testcase in a new line.
Sample Input
2 // No. of testcases
3
4
Sample Output
PRIME
NOT PRIME

Solution:

```
#include <cmath>
bool verifyPrime(int n) {
   if (n <= 1) {
      return false;
   }
   for (int i = 2; i <= sqrt(n); ++i) {
      if (n % i == 0) {
         return false;
      }
   }
   return true;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Prime factors of a number

Given an integer n, print all the prime factors of n.
For example, n = 15, its prime factors are 3 and 5
Complete the function **primeFactors()** given in the editor, that accepts an integer parameter n and prints all the prime factors of n.

Input Format

The input consists of an integer,n

Constraints

2<=n<=10^5

Output Format

Print all the prime factors of n, in new lines

Sample Input

24

Sample Output

2
2
2
3

Solution:

```
#include <cmath>
#include <cmath>
void primeFactors(int n) {
   while (n % 2 == 0) {
      std::cout << 2 << std::endl;
      n /= 2;
   }
   for (int i = 3; i <= sqrt(n); i += 2) {
      while (n % i == 0) {
         std::cout << i << std::endl;
         n /= i;
      }
   }
   if (n > 2) {
      std::cout << n << std::endl;
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Greatest Common divisor of two integers

The greatest common divisor (GCD) of two or more integers, is the largest positive integer that divides each of the integers.
For example, the GCD of 8 and 12 is 4.
Given 2 numbers , your task is to find out the GCD of the numbers.
Complete the function **gcd()** that accepts two positive non-zero integer parameters i and j and returns the greatest common divisor of these numbers.

Input Format

First line of input contains the first number, i
Second line of input contains the first number, j

Constraints

1 <= i,j <= 100000

Output Format

Print the gcd of two numbers

Sample Input

30
18

Sample Output

6

Solution:

```
int gcd(int i, int j) {
   while (j != 0) {
      int temp = j;
      j = i % j;
      i = temp;
   }
   return i;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Binary To Decimal Conversion

Binary Number System is the simplest kind of number system that uses only two digits of 0 and 1 (i.e. value of base 2). All the computers & digital electronics understand binary number system.
Now given a string consisting of digits '0' & '1', the task is to find the decimal equivalent of the number represented in the binary string.
Complete the function **binaryToDecimal** that accepts a binary string parameter whose digits are meant to represent binary (base-2) digits, and returns an integer of that number's representation in decimal (base-10).
For example, the call of **binaryToDecimal("101011")** should return **43**.
Input Format:
The first line of input contains the number of testcases,T
Then T lines follow, each line contains a string, which represents a binary number
Output Format
Print the corresponding decimal number for each testcase in a new line
Constraints:
1 <= T <= 10
It is guaranteed that the decimal number formed from the binary string will be in the range of a 32-bit integer
Do not use any built-in base conversion functions from the system libraries.
Sample Input
1
101011
Sample Output
43
Explanation:
The given binary representation is 101011,
It's decimal conversion will be 1*(2^0) + 1*(2^1) + 0*(2^2) + 1*(2^3) + 0*(2^4) + 1*(2^5)
Which will be 1 + 2 + 0 + 8 + 0 + 32 = 43

Solution:

```
#include <string>
#include <cmath>
int binaryToDecimal(std::string binary) {
    int decimal = 0;
    int length = binary.length();
    for (int i = 0; i < length; ++i) {
        if (binary[i] == '1') {
            decimal += std::pow(2, length - 1 - i);
        }
    }
    return decimal;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Functions : Addition of numbers

Write the overloaded ***sum()*** functions for addition of 2, 3 or 4 numbers passed to it.

Sample Input1

2
4 5

Sample Output1

9

Sample Input2

3
4 5 8

Sample Output2

17

Solution:

```
int sum(int a, int b) {
   return a + b;
}
int sum(int a, int b, int c) {
   return a + b + c;
}
int sum(int a, int b, int c, int d) {
   return a + b + c + d;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Functions : Display of Strings

Write the overloaded display function to display either 1 string in one line or 2 strings in one line with a hyphen '-' in between.

Sample Input1

1
CodeQuotient

Sample Output1

CodeQuotient

Sample Input2

2
CodeQuotient
coding

Sample Output2

CodeQuotient-coding

Solution:

```
#include<string>
void display(const std::string& str1) {
    std::cout << str1 << std::endl;
}
// Function to display two strings with a hyphen
void display(const std::string& str1, const std::string& str2) {
    std::cout << str1 << "-" << str2 << std::endl;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Calculate amount using compound interest

Write a program to calculate the amount using compound interest .
Input Format
Each test case contains three floating numbers denoting principle amount, rate and time.
Output Format
Print the amount upto 1(one) decimal point as shown below.
Sample Input
1000 5 2
Sample output
1102.5

Solution:

```
#include<iostream>
#include<cstdio>
#include<math.h>
#include<iomanip>
using namespace std;
int main()
{
   float principal, rate, time;
   cin >> principal >> rate >> time;
   float amount = principal * pow((1 + rate / 100), time);
   cout << fixed << setprecision(1) << amount << endl;
   return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - Rectangle

Write a class called **Rectangle** that represents a rectangular two-dimensional region. Your **Rectangle** objects should have the following methods:
public Rectangle(int x, int y, int width, int height)
Constructs a new rectangle whose top-left corner is specified by the given coordinates and with the given width and height.
public int getHeight()
Returns this rectangle's height.
public int getWidth()
Returns this rectangle's width.
public int getX()
Returns this rectangle's x-coordinate.
public int getY()
Returns this rectangle's y-coordinate.
public String toString()
Returns a string representation of this rectangle, such as ***"Rectangle[x=1,y=2,width=3,height=4]"*.**

Solution:

```
#include <string>
#include <sstream>
class Rectangle {
private:
    int x, y, width, height;
public:
    Rectangle(int x, int y, int width, int height)
        : x(x), y(y), width(width), height(height) {}
    int getHeight() const {
        return height;
    }
    int getWidth() const {
        return width;
    }
    int getX() const {
        return x;
    }
    int getY() const {
        return y;
    }
    std::string toString() const {
        std::ostringstream oss;
        oss << "Rectangle[x=" << x << ",y=" << y << ",width=" << width << ",height=" <<
height << "]";
        return oss.str();
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - TimeSpan

Define a class named TimeSpan. A TimeSpan object stores a span of time in hours and minutes (for example, the time span between 5:00am and 7:45am is 2 hours, 45 minutes). Each TimeSpan object should have the following public methods:
TimeSpan(hours, minutes)
Constructs a TimeSpan object storing the given time span of hours and minutes.
getHours()
Returns the number of hours in this time span.
getMinutes()
Returns the number of minutes in this time span, between 0 and 59.
add(hours, minutes)
Adds the given amount of time to the span. For example, (3 hours, 15 minutes) + (2 hour, 40 minutes) = (5 hours, 55 minutes). Assume that the parameters are valid: the hours are non-negative, and the minutes are between 0 and 59.
add(TimeSpan tp)
Adds the given amount of time (stored as a time span) to the current time span.
getTotalHours()
Returns the total time in this time span as the real number of hours, such as 9.75 for (9 hours, 45 min).
toString()
Returns a string representation of the time span of hours and minutes, such as "5 Hours, 38 Minutes".
Explanation:
The minutes should always be reported as being in the range of 0 to 59. That means that you may have to "carry" 60 minutes into a full hour.

Solution:

```
#include <string>
#include <sstream>
class TimeSpan {
private:
    int hours;
    int minutes;
    void normalize() {
        if (minutes >= 60) {
            hours += minutes / 60;
            minutes = minutes % 60;
        }
    }
public:
    TimeSpan(int h, int m) : hours(h), minutes(m) {
        normalize();
    }
    int getHours() const {
        return hours;
    }
    int getMinutes() const {
        return minutes;
    }
    void add(int h, int m) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        hours += h;
        minutes += m;
        normalize();
    }
    void add(const TimeSpan& tp) {
        hours += tp.hours;
        minutes += tp.minutes;
        normalize();
    }
    double getTotalHours() const {
        return hours + static_cast<double>(minutes) / 60;
    }
    std::string toString() const {
        std::ostringstream oss;
        oss << hours << " Hours, " << minutes << " Minutes";
        return oss.str();
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - Date

Write a class named **Date** that remembers information about a month and day. Ignore leap years and don't store the year in your object. You must include the following public methods:
Member-name Description
Date(int m, int d) constructs a new date representing the given month and day
daysInMonth() returns the number of days in the month stored by your date object
getDay() returns the day
getMonth() returns the month
nextDay() advances the Date to the next day, wrapping to the next month and/or year if necessary
toString() returns a string representation e.g. If date is 27 June then return "6/27"
absoluteDay() return the absolute day of a particular date e.g. January 1 is 1, February 1 is 32, March 1 is 60 and December 31 is 365.
You should define the entire class including the class heading, the private fields, and the declarations and definitions of all the public methods and constructor.

Solution:

```
#include <iostream>  // For input/output operations
#include <string>    // For string operations
#include <vector>    // For using vector
using namespace std;
class Date {
private:
   int month;
   int day;
   // Array to store the number of days in each month (ignoring leap years)
   const vector<int> daysInEachMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
   // Function to check if a given day is valid for the month
   bool isValidDate(int m, int d) {
      if (m < 1 || m > 12) return false; // Check for a valid month
      if (d < 1 || d > daysInEachMonth[m - 1]) return false; // Check for a valid day
      return true; // Return true if both month and day are valid
   }
public:
   // Constructor to initialize a Date object with month and day
   Date(int m, int d) {
      if (isValidDate(m, d)) {
         month = m;
         day = d;
      } else {
         throw invalid_argument("Invalid date!"); // Throw an error if date is invalid
      }
   }
   // Returns the number of days in the current month
   int daysInMonth() const {
      return daysInEachMonth[month - 1]; // Return days in the current month
   }
   // Returns the current day
   int getDay() const {
      return day; // Return the day of the date
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    }
    // Returns the current month
    int getMonth() const {
        return month; // Return the month of the date
    }
    // Advances the date to the next day, wrapping around months and year if needed
    void nextDay() {
        if (day < daysInMonth()) {
            day++; // Move to the next day within the same month
        } else {
            day = 1; // Reset to the first day of the next month
            if (month < 12) {
                month++; // Move to the next month
            } else {
                month = 1; // Wrap around to January if it's the end of the year
            }
        }
    }
    // Returns a string representation of the date
    string toString() const {
        return to_string(month) + "/" + to_string(day); // Format date as "month/day"
    }
    // Returns the absolute day of the year
    int absoluteDay() const {
        int absoluteDay = 0;
        for (int i = 0; i < month - 1; ++i) {
            absoluteDay += daysInEachMonth[i]; // Add up days of previous months
        }
        absoluteDay += day; // Add current day of the current month
        return absoluteDay; // Return the absolute day number
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - Circle

Write a class of objects named Circle that remembers information about a circle. You must include the following public members.
You can use the constant named Math.PI storing the value of < Â &÷Vv†Ç’ 2ã C S’à
Name Description
Circle(double r) constructs a new circle with the given radius as a real number
area() returns the area occupied by the circle (upto 2 decimal places)
circumference() returns the distance around the circle (upto 2 decimal places)
getRadius() returns the radius as a real number (upto 1 decimal places)
toString() returns a string representation such as "Circle{radius=2.5}"
You should define the entire class including the class heading, the private fields, and the declarations and definitions of all the public methods and constructor.

Solution:

```
#include <cmath>
#include <iomanip>
#include <sstream>
#include <string>
using namespace std;
class Circle {
private:
   double radius;
public:
   Circle(double r) : radius(r) {}
   double area() const {
      return round((M_PI * radius * radius) * 100) / 100;
   }
   double circumference() const {
      return round((2 * M_PI * radius) * 100) / 100;
   }
   double getRadius() const {
      return round(radius * 10) / 10;
   }
   string toString() const {
      ostringstream oss;
      oss << "Circle{radius=" << fixed << setprecision(1) << radius << "}";
      return oss.str();
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - SafeArray

One of the weakness of C++ is that it does not automatically check array indexes to see whether they are in bounds. (This makes array operations faster but less safe). We can use a class to create a safe array that checks the index of all array accesses.
Write a class called **SafeArray** that uses an int array of fixed size (say LIMIT) as its only data member, Fix it to 20 for this program. There will be two member functions.
The first, **putElement(int index,int value)**, that takes an index number and an int value as arguments and inserts the int value into the array at the index. The second,**getElement(int index)**, takes an index number as an argument and returns the int value of the element with that index. If the index is greater than LMIT , then return -1.

```
SafeArray sa1;     // define a SafeArray object
int temp = 125;   // define an int value
sa1.putElement(7, temp);     // insert value of temp into array at index 7
temp = sa1.getElement(7);   // obtain value from array at index 7
```

Both functions should check the index argument to make sure it is not less than 0 or greater than LIMIT-1.

Input Format

The first line contains an integer, n, the denoting number of elements.
The next 2*N contains and integer, value and index , in separate lines

Output Format

If the index is a valid index, print its content , else print "Out Of Bounds".

Sample Input

```
2
125
7
135
31
```

Sample Output

```
125
Out of Bounds
```

Explanation:

First line of input denotes the number of elements. Next each two lines denotes the number and its index in array.
Print the number in new line after inserting it in the array.

Solution:

```
#include<iostream>
using namespace std;
class SafeArray {
private:
   int* arr;
   int size;
public:
   // Constructor that initializes the array with a given size
   SafeArray(int s) : size(s) {
      arr = new int[size];
      for (int i = 0; i < size; ++i) {
         arr[i] = 0;
      }
   }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    // Destructor to release allocated memory
    ~SafeArray() {
        delete[] arr;
    }
    // Function to insert a value into the array at a specific index
    void putElement(int index, int value) {
        if (index >= 0 && index < size) {
            arr[index] = value;
        }
    }
    // Function to retrieve a value from the array at a specific index
    int getElement(int index) const {
        if (index >= 0 && index < size) {
            return arr[index];
        } else {
            return -1;  // Return -1 if the index is out of bounds
        }
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - TollBooth

Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay 50 Rupees toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected.
Model this tollbooth with a class called TollBooth. The two data items are to hold the total number of cars, and to hold the total amount of money collected.
A constructor initializes both of these to 0.A member function called payingCar() increments the car total and adds 50/- to the cash total.Another function, called nopayCar(), increments the car total but adds nothing to the cash total.Finally, a member function called display() displays the two totals.
Make appropriate member functions const.
The main function should allow the user to press 'p' to count a paying car, and 'n' to count a nonpaying car. Pushing the 'q' key should cause the program to print out the total cars and total cash and then exit.

Sample Input

p
p
p
n
p
n
p
q

Sample Output

Total Cash : 250/-
Total Cars : 7

Solution:

```
#include <iostream>
class TollBooth {
private:
   int totalCars;
   int totalCash;
public:
   // Constructor
   TollBooth() : totalCars(0), totalCash(0) {}
   // Member functions
   void payingCar() {
      totalCars++;
      totalCash += 50;
   }
   void nopayCar() {
      totalCars++;
   }
   void display() const {
      std::cout << "Total Cash : " << totalCash << "/-" << std::endl;
      std::cout << "Total Cars : " << totalCars << std::endl;
   }
};
int main() {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    TollBooth booth;
    char input;
    while (true) {
        std::cin >> input;
        if (input == 'p') {
            booth.payingCar();
        } else if (input == 'n') {
            booth.nopayCar();
        } else if (input == 'q') {
            booth.display();
            break;
        } else {
            std::cout << "Invalid input. Please enter 'p', 'n', or 'q'." << std::endl;
        }
    }
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Count number of Objects

Create a class for counting the number of objects created and destroyed within various block using constructor and destructors.

Solution:

```
int cnt = 0;
class Counter
{
public:
   Counter()
   {
      cnt++;
   }
   ~Counter()
   {
      cnt--;
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: class - Box

Design a class named *Box* whose dimensions are integers and private to the class.
The dimensions are labelled: length, breadth and height.
The default constructor of the class should initialize them to 0 (ZERO).The parameterized constructor *Box(int length, int breadth, int height)* should initialize *them* to length, breadth and height respectively.The copy constructor *Box(Box* b1) should set them to b1's length, breadth and height respectively.
Apart from the above, the class should have functions:
int getLength() - Return box's length*int getBreadth()* - Return box's breadth*int getHeight()* - Return box's height*long long CalculateVolume()* - Return the volume of the box

Solution:

```
class Box {
private:
   int length, breadth, height;
public:
   Box() : length(0), breadth(0), height(0) {}
   Box(int length, int breadth, int height) : length(length), breadth(breadth), height(height) {}
   Box(const Box& b) : length(b.length), breadth(b.breadth), height(b.height) {}
   int getLength() const { return length; }
   int getBreadth() const { return breadth; }
   int getHeight() const { return height; }
   long long CalculateVolume() const {
      return (long long)length * breadth * height;
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Polymorphism : Area

Calculate the volume of geometrical shapes using virtual functions for below shapes. Use the following formulas and display the volume upto 3 decimal places.
Volume of sphere= 4/3*3.14*radius*radius*radiusVolume of cylinder=3.14*radius*radius*heightVolume of cone=1/3*3.14*radius*radius*height
Volume of cube= side*side*side
Choice 1 for sphere, 2 for cylinder, 3 for cone, 4 for cube
Sample input 1
1 //choice for sphere
3 //radius of sphere
Sample output 1
Volume of sphere is 112.757
Sample input 2
2 //choice for cylinder
3 //radius of cylinder
4 //height of cylinder
Sample output 2
Volume of cylinder is 113.040
Sample input 3
5 // wrong choice
Sample output 3
wrong choice
Note: First line of input contain choice. Second line contain radius/side . For Cylinder and Cone third line contain height.

Solution:

```
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;
int main() {
   int choice;
   cin >> choice;
   if (choice == 1) { // Sphere
      double radius;
      cin >> radius;
      double volume =  1.33*3.14*radius*radius*radius;
      printf("Volume of sphere is %.3f\n", volume);
   } else if (choice == 2) { // Cylinder
      double radius, height;
      cin >> radius >> height;
      double volume = 3.14*radius*radius*height;
      printf("Volume of cylinder is %.3f\n", volume);
   } else if (choice == 3) { // Cone
      double radius, height;
      cin >> radius >> height;
      double volume = 0.33*3.14*radius*radius*height;
      printf("Volume of cone is %.3f\n", volume);
   } else if (choice == 4) { // Cube
      double side;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        cin >> side;
        double volume = side*side*side;
        printf("Volume of cube is %.3f\n", volume);
    } else { // Wrong choice
        cout << "wrong choice" << endl;
    }
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: List as Stack or Queue

Create a class List with two pure virtual function store() and retrieve().
To store a value call store and
To retrieve call retrieve function.
Derive two classes Stack and Queue from it and override store and retrieve methods appropriately.

Sample input1

```
1  // for Stack
1  // for store
5
1  // for store
6
1  // for store
7
2  // for retrieve
2  // for retrieve
2  // for retrieve
```

Sample output1

```
7
6
5
```

Sample input2

```
1  // for Queue
1  // for store
5
1  // for store
6
1  // for store
7
2  // for retrieve
2  // for retrieve
2  // for retrieve
```

Sample output2

```
5
6
7
```

Solution:

```
#include<iostream>
#include<vector>
using namespace std;
// Base class List with pure virtual functions
class List {
public:
    virtual void store(int value) = 0;
    virtual int retrieve() = 0;
};
// Derived class Stack (LIFO behavior)
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
class Stack : public List {
private:
    vector<int> stack;
public:
    void store(int value) override {
        stack.push_back(value);
    }
    int retrieve() override {
        if (!stack.empty()) {
            int value = stack.back();
            stack.pop_back();
            return value;
        }
        return -1; // Return -1 or some error code if the stack is empty
    }
};
// Derived class Queue (FIFO behavior)
class Queue : public List {
private:
    vector<int> queue;
public:
    void store(int value) override {
        queue.push_back(value);
    }
    int retrieve() override {
        if (!queue.empty()) {
            int value = queue.front();
            queue.erase(queue.begin());
            return value;
        }
        return -1; // Return -1 or some error code if the queue is empty
    }
};
int main() {
    int choice;
    cin >> choice;
    List* list;
    if (choice == 1) {
        list = new Stack();
    } else if (choice == 2) {
        list = new Queue();
    } else {
        return 0;  // Exit if the choice is not 1 or 2
    }
    int operation;
    while (cin >> operation) {
        if (operation == 1) {
            int value;
            cin >> value;
            list->store(value);
        } else if (operation == 2) {
            int result = list->retrieve();
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            if (result != -1) {
                cout << result << endl;
            }
        }
    }
    delete list;
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Person Professor and Student

Create three classes **Person, Professor** and **Student**.
The class Person should have data member age.The classes Professor and Student should inherit from the class Person.The class Professor should have two integer members: publications and cur_id. There will be two member functions: getdata and putdata. The function getdata should get the input from the user: the age and publications of the professor. The function putdata should print the age, publications and the cur_id of the professor.The class Student should have two data members: marks, which is an array of size 6 and cur_id. It has two member functions: getdata and putdata. The function getdata should get the input from the user: the age, and the marks of the student in subjects. The function putdata should print the age, sum of the marks and the cur_id of the student.For each object being created of the Professor or the Student class, sequential id's should be assigned to them starting from 1.

Solve this problem using virtual functions, constructors and static variables. You can create more data members if you want.

Solution:

```
#include<iostream>
using namespace std;
// Base class Person
class Person {
protected:
    int age;
public:
    virtual void getdata() = 0;
    virtual void putdata() = 0;
};
// Derived class Professor
class Professor : public Person {
private:
    int publications;
    int cur_id;
    static int id_counter;  // Static variable to keep track of the IDs
public:
    Professor() {
        cur_id = ++id_counter;  // Assign unique ID and increment the counter
    }
    void getdata() override {
        cin >> age >> publications;
    }
    void putdata() override {
        cout << age << " " << publications << " " << cur_id << endl;
    }
};
int Professor::id_counter = 0;  // Initialize static ID counter
// Derived class Student
class Student : public Person {
private:
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    int marks[6];
    int cur_id;
    static int id_counter;  // Static variable to keep track of the IDs
public:
    Student() {
        cur_id = ++id_counter;  // Assign unique ID and increment the counter
    }
    void getdata() override {
        cin >> age;
        for (int i = 0; i < 6; ++i) {
            cin >> marks[i];
        }
    }
    void putdata() override {
        int sum_marks = 0;
        for (int i = 0; i < 6; ++i) {
            sum_marks += marks[i];
        }
        cout << age << " " << sum_marks << " " << cur_id << endl;
    }
};
int Student::id_counter = 0;  // Initialize static ID counter
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Cut the sticks

Given an array, ***lengths***, of N stick lengths (where each length is a positive integer), a cut operation reduces the length of each stick in the array by the length of the array's shortest stick. A stick can only be cut if it has a length "e 1.
Complete the **cutSticks()** function which has 3 parameters:
Size of the array (lengths_size).An integer array of stick lengths (lengths). Pointer to the variable which is used to store the size of the result array.
Your function must perform cut operations on lengths until every stick length is reduced to 0 and return an integer array, where ith element of the array denotes the individual sticks cut during the ith operation.
Your function must return an integer array, where ith element of the array denotes the individual sticks cut during the ith operation.

Input Format

First line contains an integer i.e. N denoting number of sticks.
Next N lines, contains an integer each denoting the length of ith stick.

Output Format

Print the number of sticks which have a cut during ith cut operation in new lines.

Sample Input 1

```
6
5
4
4
2
2
8
```

Sample Output 1

```
6
4
2
1
```

Explanation 1

|lengths| = 6, lengths = {5, 4, 4, 2, 2, 8}
Cut Operation 1: The smallest length in lengths is 2, so the cut length for this cut operation is 2.

After cutting (reducing) each stick in lengths by 2, lengths1 = {3, 2, 2, 0, 0, 6} and we print 6 (the number of sticks cut during the cut operation) on a new line.
Cut Operation 2: The smallest length in lengths1 = {3, 2, 2, 0, 0, 6} is 2, so the cut length for this cut operation is 2.

After cutting (reducing) each stick in lengths1 by 2, lengths2 = {1, 0, 0, 0, 0, 4} and we print 4(the number of sticks cut during the cut operation) on a new line.
Cut Operation 3: The smallest length in lengths2 = {1, 0, 0, 0, 0, 4} is 1, so the cut length for this cut operation is 1.

After cutting (reducing) each stick in lengths2 by 1, lengths3 = {0, 0, 0, 0, 0, 3} and we print 2(the number of sticks cut during the cut operation) on a new line.
Cut Operation 4: The smallest length in lengths3 = {0, 0, 0, 0, 0, 3} is 3, so the cut length for this cut operation is 3.

After cutting (reducing) each stick in lengths3 by 3, lengths4 = {0, 0, 0, 0, 0, 0} and we print 1(the number of sticks cut during the cut operation) on a new line.
At this point, there are no more sticks to be cut and we cease performing cut operations.

ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Sample Input 2

8
1
2
3
4
3
3
2
1

Sample Output 2

8
6
4
1

Solution:

```
int* cutSticks(int lengths_size, int *lengths, int *result_size)
{
int* result_array = new int[lengths_size];
    int result_index = 0;
    while (true) {
        int min_length = -1;
        int sticks_cut = 0;
        // Find the minimum non-zero length
        for (int i = 0; i < lengths_size; i++) {
            if (lengths[i] > 0 && (min_length == -1 || lengths[i] < min_length)) {
                min_length = lengths[i];
            }
        }
        // If no minimum is found, break the loop (all sticks are zero)
        if (min_length == -1) {
            break;
        }
        // Perform the cut and count how many sticks are cut
        for (int i = 0; i < lengths_size; i++) {
            if (lengths[i] > 0) {
                lengths[i] -= min_length;
                sticks_cut++;
            }
        }
        // Store the result of this operation
        result_array[result_index++] = sticks_cut;
    }
    // Update the result size
    *result_size = result_index;
    return result_array;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Merge two Arrays

Write a function **mergeArrays** which should merge two sorted arrays to create one single sorted array.

Complete the function **int* mergeArrays(int a[], int b[] , int asize, int bsize)** below which takes pointers to the first element of the two sorted arrays & the size of the arrays and return the base address of the final sorted array.

Sample Input

```
4       // Size of 1st array
1 2 3 6    // Elements of 1st array
3       // Size of 2nd array
4 5 7      // Elements of 2nd array
```

Sample Output

```
1
2
3
4
5
6
7
```

Solution:

```
int * mergeArrays(int a[], int b[], int asize, int bsize) {
  int* mergedArray = new int[asize + bsize];
    int i = 0, j = 0, k = 0;
    // Merge the two arrays
    while (i < asize && j < bsize) {
        if (a[i] <= b[j]) {
            mergedArray[k++] = a[i++];
        } else {
            mergedArray[k++] = b[j++];
        }
    }
    // Copy remaining elements of array a, if any
    while (i < asize) {
        mergedArray[k++] = a[i++];
    }
    // Copy remaining elements of array b, if any
    while (j < bsize) {
        mergedArray[k++] = b[j++];
    }
    // Return the base address of the merged array
    return mergedArray;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Copy the array Deep

Complete the below class as deep copy of array defined using copy constructor and assignment operator.

Solution:

```
class Array {
private:
    int *arr;
public:
    // Default constructor
    Array() {
        arr = new int[4];
        for (int i = 0; i < 4; i++) {
            arr[i] = i;
        }
    }
    // Copy constructor for deep copy
    Array(const Array &other) {
        arr = new int[4]; // Allocate new memory
        for (int i = 0; i < 4; i++) {
            arr[i] = other.arr[i]; // Copy the data
        }
    }
    // Assignment operator for deep copy
    Array& operator=(const Array &other) {
        if (this == &other) {
            return *this; // Handle self-assignment
        }
        // Deallocate the existing memory
        delete[] arr;
        // Allocate new memory and copy the data
        arr = new int[4];
        for (int i = 0; i < 4; i++) {
            arr[i] = other.arr[i];
        }
        return *this;
    }
    // Destructor to free the allocated memory
    ~Array() {
        delete[] arr;
    }
    // Method to print the array
    void printArray() {
        for (int i = 0; i < 4; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
    // Method to change the array's elements
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    void changeArray() {
        for (int i = 0; i < 4; i++) {
            arr[i] = i * i;
        }
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - ArrayClass

You might have seen a class that acts like an array. Following is an implementation given for it.
Your task is to add an overloaded assignment operator and an overloaded copy constructor to the Array class.
So that statements such as
ArrayClass arr2(arr1);
and
arr3 = arr1;
must work accordingly.

Solution:

```
class ArrayClass
{
private:
   int* ptr;  // pointer to Array contents
   int size;  // size of Array
public:
   // One-argument constructor
   ArrayClass(int s)
   {
      size = s;            // argument is size of Array
      ptr = new int[s];      // make space for Array
   }
   // Destructor
   ~ArrayClass()
   {
      delete[] ptr;
   }
   // Copy Constructor
   ArrayClass(const ArrayClass& other)
   {
      size = other.size;
      ptr = new int[size];  // Allocate new memory
      for (int i = 0; i < size; i++) {
         ptr[i] = other.ptr[i];  // Copy elements
      }
   }
   // Assignment Operator
   ArrayClass& operator=(const ArrayClass& other)
   {
      if (this == &other) {
         return *this;  // Handle self-assignment
      }
      delete[] ptr;  // Free existing memory
      size = other.size;
      ptr = new int[size];  // Allocate new memory
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        for (int i = 0; i < size; i++) {
            ptr[i] = other.ptr[i];  // Copy elements
        }
        return *this;
    }
    // Overloaded subscript operator
    int& operator[](int j)
    {
        return *(ptr + j);
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - ComplexNumber

Implements Complex numbers using classes in C++.
The program implements class Complex which gives facility to initialize Complex objects using constructors and add and subtract two Complex objects using + and - operators.
Also add one display() function to display the complex number as **12+4i** format, where 12 is real part and 4 is imaginary part of it.

Solution:

```
#include <iostream>
using namespace std;
class Complex
{
private:
   int real, imag;
public:
   // Default constructor
   Complex() : real(0), imag(0) {}
   // Parameterized constructor
   Complex(int r, int i) : real(r), imag(i) {}
   // Overload + operator for addition
   Complex operator+(const Complex &other) const {
      return Complex(real + other.real, imag + other.imag);
   }
   // Overload - operator for subtraction
   Complex operator-(const Complex &other) const {
      return Complex(real - other.real, imag - other.imag);
   }
   // Display function
   void display() const {
      cout << real;
      if (imag >= 0)
         cout << "+" << imag << "i";
      else
         cout << imag << "i";
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - DistanceCalculator

Create a Distance class, which can measure distance in feet and inches as defined below: readDistance() method to read two double numbers, first the feet and second the inches. display() function to display the feet and inches using notations e.g. 2 feets and 5 inches should be printed as 2'5"Add an overloaded plus "+" operator that adds two distances. It should allow statements like dist3 = dist1 + dist2;Add an overloaded minus "-" operator that subtracts two distances. It should allow statements like dist3 = dist1 - dist2; Assume that the operator will never be used to subtract a larger number from a smaller one (that is, negative distances are not allowed).

Solution:

```
#include <iostream>
using namespace std;
class Distance
{
private:
   double feet, inch;
public:
   // Default constructor
   Distance() : feet(0), inch(0) {}
   // Parameterized constructor
   Distance(double f, double i) : feet(f), inch(i) {
      normalize();  // Normalize the distance to keep inches between 0 and 11
   }
   // Method to read distance
   void readDistance() {
      cin >> feet >> inch;
      normalize();  // Normalize after reading input
   }
   // Method to display distance
   void display() const {
      cout << static_cast<int>(feet) << "'"
           << static_cast<int>(inch) << "\"";
   }
   // Overload + operator for addition
   Distance operator+(const Distance &other) const {
      double totalFeet = feet + other.feet;
      double totalInches = inch + other.inch;
      // Normalize result
      return Distance(totalFeet + totalInches / 12, fmod(totalInches, 12));
   }
   // Overload - operator for subtraction
   Distance operator-(const Distance &other) const {
      double totalFeet = feet - other.feet;
      double totalInches = inch - other.inch;
      // Normalize result
      if (totalInches < 0) {
         totalFeet -= 1;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            totalInches += 12; // Add 12 inches to make inches positive
        }
        return Distance(totalFeet, totalInches);
    }
private:
    // Normalize inches to be within 0 and 11
    void normalize() {
        if (inch >= 12) {
            feet += inch / 12;
            inch = fmod(inch, 12);
        } else if (inch < 0) {
            feet -= (abs(inch) / 12) + 1;
            inch = 12 + fmod(inch, 12);
        }
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - GoodString

The String defined as character array has a flaw: It does not protect itself if it is initialized with too many characters. For example, the definition
String s = “This string will surely exceed the width of the string constant defined, so it is an example of bad string.”;
will cause the array in s to overflow, with unpredictable consequences, such as crashing the system.
Define a class GoodString that prevents buffer overflow when too long a string constant is used in a definition. A new constructor in the derived class should copy only a FIXED number of characters into str if the string constant is longer, but copy the entire constant if it’s shorter. In this new class incorporate three new functions: left(), mid(), and right().
s2 = s1.left(n) // s2 is assigned the leftmost n characters from s1
s2 = s1.mid(s, n) // s2 is assigned the middle n characters from s1, starting at character number s(leftmost character is 0)
s2 = s1.right(n) // s2 is assigned the rightmost n characters from s1
Sample Input1
20
Code Quotient - Get better at Coding
Sample Output1
Code Quotient - Get
Sample Input2
60
Code Quotient - Get better at Coding
Sample Output2
Code Quotient - Get better at Coding

Solution:

```
#include<iostream>
#include<cstring>
using namespace std;
class GoodString {
private:
   static const int MAX_SIZE = 100; // Fixed buffer size
   char str[MAX_SIZE + 1];          // +1 for null terminator
public:
   // Constructor
   GoodString(const char* input) {
      // Copy only up to MAX_SIZE characters
      strncpy(str, input, MAX_SIZE);
      str[MAX_SIZE] = '\0'; // Ensure null termination
   }
   // Function to return the leftmost n characters
   GoodString left(int n) {
      char temp[MAX_SIZE + 1];
      strncpy(temp, str, n);
      temp[n] = '\0';
      return GoodString(temp);
   }
   // Function to return n characters starting from index s
   GoodString mid(int s, int n) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        char temp[MAX_SIZE + 1];
        strncpy(temp, str + s, n);
        temp[n] = '\0';
        return GoodString(temp);
    }
    // Function to return the rightmost n characters
    GoodString right(int n) {
        int len = strlen(str);
        char temp[MAX_SIZE + 1];
        strncpy(temp, str + len - n, n);
        temp[n] = '\0';
        return GoodString(temp);
    }
    // Function to display the string
    void display() {
        cout << str << endl;
    }
};
int main() {
    // Sample Input
    int n;
    cin >> n;
    cin.ignore();
    char input[200];
    cin.getline(input, 200);
    GoodString s1(input);
    GoodString s2 = s1.left(n);
    // Sample Output
    s2.display();
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - IntValue

Create a class IntValue. Include overloaded member function setValue() to initialize an IntValue to 0, to an given parameter value. Also add one display() method to display the IntValue.
Overload three integer arithmetic operators (+, - and *) so that they operate on objects of type IntValue.
If the result of any such arithmetic operation exceeds the normal range of ints (in a 32-bit environment)—from 2,147,483,648 to –2,147,483,647—have the operator print a warning "Out of Range" and terminate the program.
Hint: To facilitate checking for overflow, perform the calculations using type long double.

Solution:

```
class IntValue {
private:
   int value;
public:
   IntValue() : value(0) {}
   IntValue(int a) : value(a) {}
   void setValue() {
      value = 0;
   }
   void setValue(int a) {
      value = a;
   }
   void display() {
      cout << value;
   }
   IntValue operator+(const IntValue& other) const {
      long long result = static_cast<long long>(value) + static_cast<long long>(other.value);
      if (result < -2147483647 || result > 2147483647) {
         cout << "Out of Range" << endl;
         exit(1);
      }
      return IntValue(static_cast<int>(result));
   }
   IntValue operator-(const IntValue& other) const {
      long long result = static_cast<long long>(value) - static_cast<long long>(other.value);
      if (result < -2147483647 || result > 2147483647) {
         cout << "Out of Range" << endl;
         exit(1);
      }
      return IntValue(static_cast<int>(result));
   }
   IntValue operator*(const IntValue& other) const {
      long long result = static_cast<long long>(value) * static_cast<long long>(other.value);
      if (result < -2147483647 || result > 2147483647) {
         cout << "Out of Range" << endl;
         exit(1);
      }
      return IntValue(static_cast<int>(result));
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Class - Matrix

Create a 'MATRIX' class of size m X n. Overload the '+' operator to add two MATRIX objects. m and n will be inputs followed by m*n elements.

Sample input

2 2
3 3 3 3
4 4 4 4

Sample output

7 7 7 7

Solution:

```
#include <iostream>
using namespace std;
class Matrix
{
   int m, n, a[20][20]; // Private members to store matrix dimensions and elements
public:
   // Parameterized Constructor: Initializes m and n
   Matrix(int x, int y)
   {
      m = x;
      n = y;
      // Optionally initialize the matrix elements to 0
      for(int i = 0; i < m; i++)
         for(int j = 0; j < n; j++)
            a[i][j] = 0;
   }
   // Function to read matrix elements from input
   void readmat()
   {
      for(int i = 0; i < m; i++)
         for(int j = 0; j < n; j++)
            cin >> a[i][j];
   }
   // Overloading the '+' operator to add two Matrix objects
   Matrix operator +(Matrix a2)
   {
      Matrix res(m, n); // Resultant matrix with the same dimensions
      for(int i = 0; i < m; i++)
         for(int j = 0; j < n; j++)
            res.a[i][j] = a[i][j] + a2.a[i][j]; // Element-wise addition
      return res; // Return the resultant matrix
   }
   // Function to display matrix elements in a single line separated by spaces
   void display()
   {
      for(int i = 0; i < m; i++)
         for(int j = 0; j < n; j++)
            cout << a[i][j] << " "; // Print each element followed by a space
      cout << endl; // Newline after printing all elements
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: class - Box2

Design a class named *Box2* whose dimensions are integers and private to the class.
The dimensions are labelled: length, breadth and height.
The default constructor of the class should initialize them to 0 (ZERO).The parameterized constructor *Box2(int length, int breadth, int height)* should initialize *them* to length, breadth and height respectively.The copy constructor *Box2(Box2* b1) should set them to b1's length, breadth and height respectively.
Apart from the above, the class should have functions:
int getLength() - Return box's length*int getBreadth()* - Return box's breadth*int getHeight()* - Return box's height*long CalculateVolume()* - Return the volume of the box

þÿOverload the operator < (less than) for the class *Box2*. *object b1 is < object b2* if:
b1.length < b2.length and b1.breadth < b2.breadth and b1.height < b2.height
þÿOverload operator << for the class *Box2()*.
If b1 is an object of class *Box*:
cout<<b1 should print length, breadth and height on a single line separated by spaces.

Solution:

```
#include <iostream>
using namespace std;
class Box2 {
private:
    int length, breadth, height;
public:
    // Default constructor
    Box2() : length(0), breadth(0), height(0) {}
    // Parameterized constructor
    Box2(int l, int b, int h) : length(l), breadth(b), height(h) {}
    // Copy constructor
    Box2(const Box2 &b1) : length(b1.length), breadth(b1.breadth), height(b1.height) {}
    // Getter functions
    int getLength() const {
        return length;
    }
    int getBreadth() const {
        return breadth;
    }
    int getHeight() const {
        return height;
    }
  // Function to calculate volume
    long long CalculateVolume() const {
        return static_cast<long long>(length) * breadth * height;
    }
    // Overloading the '<' operator
    bool operator<(const Box2 &b) const {
        return (length < b.length && breadth < b.breadth && height < b.height);
    }
    // Overloading the '<<' operator
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    friend ostream& operator<<(ostream &out, const Box2 &B) {
        out << B.length << " " << B.breadth << " " << B.height;
        return out;
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : Actors

Design a class Person and two sub classes Actor and Actress. Every Actor will have a name, color, number_of_eyes & debut_year associated with him/her.
Supply each with a constructor that sets the Person data û elds as described with given values, and a method named **toString()** that returns a string that contains a complete description of the Actor.
The constructor should be in the following format
Actor(name,color,number_of_eyes,debut_year)
And the same for class **Actress.**

Sample input

Amitabh
BROWN
2
1965
Hema
White
2
1975

Sample output

The person Amitabh is an Actor. He is BROWN in color, has 2 eyes and debut in 1965
The person Hema is an Actress. She is White in color, has 2 eyes and debut in 1975

Solution:

```
#include <iostream>
#include <string>
using namespace std;
class Person {
public:
   string name;
   string color;
   int number_of_eyes;
   int debut_year;
   Person(string name, string color, int number_of_eyes, int debut_year) : name(name),
color(color), number_of_eyes(number_of_eyes), debut_year(debut_year) {}
};
class Actor : public Person {
public:
   Actor(string name, string color, int number_of_eyes, int debut_year) : Person(name, color,
number_of_eyes, debut_year) {}
   string toString() {
      return "The person " + name + " is an Actor. He is " + color + " in color, has " +
to_string(number_of_eyes) + " eyes and debut in " + to_string(debut_year) + "\n";
   }
};
class Actress : public Person {
public:
   Actress(string name, string color, int number_of_eyes, int debut_year) : Person(name,
color, number_of_eyes, debut_year) {}
   string toString() {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        return "The person " + name + " is an Actress. She is " + color + " in color, has " +
to_string(number_of_eyes) + " eyes and debut in " + to_string(debut_year) + "\n";
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : CalculateBill

Design a solution that has following features:
Generate bill on the basis of Item price and quantity i.e. **Bill = price of item * quantity**
Calculates cash from notes of Rs 2000, Rs 500, Rs 100, Rs 50, and Rs 10.Match cash against bill and display “Clear” message if no balance was there otherwise print the amount needs to pay.

Develop two classes Bill and Cash, where Cash inherits Bill. Sample input and output are shown below:
Sample Input 1:
1000 //item_price
100 //qty
4 //notes of 2000
0 //notes of 500
0 //notes of 100
0 //notes of 50
10 //notes of 10
Sample Output 1:
Need to pay: 91900
Constraints: notes should be of Rs 2000, Rs 500, Rs 100, Rs 50, and Rs 10.
Formula Used: Bill = price of item * quantity
by

Solution:

```
#include<iostream>
using namespace std;
class Bill {
protected:
   int item_price;
   int qty;
   int total_bill;
public:
   void calculate_bill() {
      total_bill = item_price * qty;
   }
   void set_item_price(int price) {
      item_price = price;
   }
   void set_quantity(int quantity) {
      qty = quantity;
   }
   int get_total_bill() {
      return total_bill;
   }
};
class Cash : public Bill {
private:
   int notes_2000;
   int notes_500;
   int notes_100;
   int notes_50;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    int notes_10;
    int total_cash;
public:
    void get_cash() {
        // Input the item price and quantity
        int price, quantity;
        cin >> price;
        cin >> quantity;
        set_item_price(price);
        set_quantity(quantity);
        // Calculate the bill
        calculate_bill();
        // Input the number of notes
        cin >> notes_2000;
        cin >> notes_500;
        cin >> notes_100;
        cin >> notes_50;
        cin >> notes_10;
        // Calculate total cash from notes
        total_cash = (notes_2000 * 2000) + (notes_500 * 500) + (notes_100 * 100) +
                (notes_50 * 50) + (notes_10 * 10);
    }
    void display() {
        int amount_due = get_total_bill() - total_cash;
        if (amount_due <= 0) {
            cout << "Clear" << endl;
        } else {
            cout << "Need to pay: " << amount_due << endl;
        }
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : BookCD

Imagine a publishing company that markets both Book and CD's of its works. A class Publication is created that stores the title (a string) and price (type int) of a publication. From this class derive two classes:
Book, which adds a page count and author, and a constructor having title, price, pages & writer as parameters.CD, which adds a playing time in minutes, and a constructor having title, price & length as parameters.
Each of these two classes should have a putdata() function to display its data as shown in sample output.

Sample Input

```
Programming-In-C    // Title of Book
150        // Price of Book
1025        // Pages of Book
Schildt        // Writer of Book
Rock-On      // Title of CD
50        // Price of CD
185          // Length of CD
```

Sample Output

```
Book Title "Programming-In-C", written by "Schildt" has 1025 pages and of 150 rupees.
CD Title "Rock-On", is of 185 minutes length and of 50 rupees.
```

Solution:

```
class Book : public Publication
{
private:
   int pages;
   string author;
public:
   Book(string t, int p, int pg, string a) {
      title = t;
      price = p;
      pages = pg;
      author = a;
   }
   void putdata() const {
      cout << "Book Title \"" << title << "\", written by \"" << author
           << "\" has " << pages << " pages and of " << price << " rupees." << endl;
   }
};
class CD : public Publication
{
private:
   int playingTime;
public:
   CD(string t, int p, int pt) {
      title = t;
      price = p;
      playingTime = pt;
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    }
    void putdata() const {
        cout << "CD Title \"" << title << "\", is of " << playingTime
             << " minutes length and of " << price << " rupees." << endl;
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : FilteredAccount

A cash processing company has a class called **Account** used to process transactions:
Method/Constructor Description
Account(int accno) **constructs an account using account number given**
boolean process(Transaction t) **processes the next transaction, returning true if transaction was approved, false otherwise**
Account objects interact with **Transaction** objects, which have many methods including:
int value() **returns the value of this transaction in rupees (could be negative, positive or zero)**
The company wishes to create a slight modification to the Account class that filters out zero-valued transactions. Design a new class called **FilteredAccount** whose instances can be used in place of an **Account** object but which include the extra behavior of not processing transactions with a value of 0. More specifically, the new class should indicate that a zero-valued transaction was approved but shouldn't call the process method in the **Account** class to process it. Your class should have a single constructor that accepts a parameter of type int, and it should include the following method:
int filtered() **returns the number of transactions filtered out; returns 0 if no transaction submitted**

(**Hint**: override the process() method and call Account class process() for non-zero value transaction, otherwise ignore it. Also put a counter for transactions and manipulate in process() method.)

Solution:

```
class FilteredAccount : public Account {
private:
   int filteredCount;
public:
   FilteredAccount(int accno) : Account(accno), filteredCount(0) {}
   bool process(Transaction t) {
      if (t.value() == 0) {
         filteredCount++;
         return true;
      } else {
         return Account::process(t);
      }
   }
   int filtered() const {
      return filteredCount;
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : MemoryCalculator

A class Calculator that performs calculations on integers is defined as below: -

Member	Description
Calculator(int seed)	**constructs a Calculator with given seed for random numbers**
boolean isPrime(int n)	**returns true if n is prime**
int kthPrime(int k)	**returns the kth prime (assumes k >= 1)**
int fib(int n)	**returns the nth Fibonacci number (assumes n >= 1)**
int rand(int max)	**returns a random value between 0 and max**

The class correctly computes its results, but it does so inefficiently. In particular, it often computes the same value more than once. You are to implement a technique known as "memoizing" to speed up the computation of primes. The idea behind memoizing is to remember values that have been computed previously. For example, suppose that the value kthPrime(30) is requested 100 times. There is no reason to compute it 100 different times. Instead you can compute it once and store its value, so that the 99 calls after the first simply return the "memoized" value (the remembered value).

Define a new class called MemoryCalculator that can be used in place of a Calculator to speed up the prime computation. A MemoryCalculator object should behave just like a Calculator object except that it should guarantee that the value of kthPrime(k) is computed only once for any given value k. Your class should still rely on the Calculator class to compute each value for kthPrime(k). It is simply guaranteeing that the computation is not performed more than once for any particular value of k. The isPrime method calls kthPrime, so it does not need to be memoized. You do not need to memoize the Fibonacci computation. You should not make any assumptions about how large k might be or about the order in which the method is called with different values of k.

Your class should also provide the following public member functions that will allow a client to find out how many values have been directly computed versus how many calls have been handled through memoization.

Member	Description
MemoryCalculator(int seed)	**constructs a MemoryCalculator with given seed for random numbers**
int getComputeCount()	**returns number of values actually computed**
int getMemoCount()	**returns number of calls handled through memoization**

Sample Input

```
3  // seed for random numbers
3  // No. of time the kthPrime() function will be called
7  // Parameter for kthPrime() function
```

Sample Output

```
1 // Number of times kthPrime() is called
2 // Number of times kthPrime() result is fetched from memory
```

Solution:

```
#include <iostream>
#include <unordered_map>
class MemoryCalculator : public Calculator {
private:
   std::unordered_map<int, int> memo;
   int computeCount;
   int memoCount;
public:
   MemoryCalculator(int seed) : Calculator(seed), computeCount(0), memoCount(0) {}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
    int kthPrime(int k) {
        if (memo.find(k) != memo.end()) {
            ++memoCount;
            return memo[k];
        } else {
            ++computeCount;
            int result = Calculator::kthPrime(k);
            memo[k] = result;
            return result;
        }
    }
    int getComputeCount() const {
        return computeCount;
    }
    int getMemoCount() const {
        return memoCount;
    }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : Dice

A ***Dice*** class representing a set of 6-sided dice that can be rolled by a player, is to be implemented as below:

Member Description

private: int* diceValues **an array of all dice values rolled**
private: int count **length of diceValues array**
Dice(int count) **constructs a dice roller to roll the given # of dice; all dice initially have the value of 6**
virtual int getCount() const **returns the number of dice managed by this dice roller, as passed to the constructor**
virtual int getValue(int index) const **returns the die value (1-6) at the given 0-based index**
virtual void roll(int index) **rolls the given die to give it a new random value from 1-6**
virtual int total() const **returns the sum of all current dice values in this dice roller**
virtual string toString() const **returns string of dice values, e.g. "{4, 1, 6, 5}"**
ostream& operator <<(ostream& out, Dice& dice) **prints the given dice in its toString format**

Solution:

```
#include <iostream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <ctime>
class Dice {
public:
   // Constructor
   Dice(int count) : count(count) {
      diceValues = new int[count];
      for (int i = 0; i < count; ++i) {
         diceValues[i] = 6; // Initialize all dice values to 6
      }
   }
   // Destructor
   ~Dice() {
      delete[] diceValues; // Clean up dynamically allocated memory
   }
   // Returns the number of dice
   virtual int getCount() const {
      return count;
   }
   // Returns the value of the die at the given index
   virtual int getValue(int index) const {
      if (index >= 0 && index < count) {
         return diceValues[index];
      }
      return -1; // Invalid index
   }
   // Rolls the die at the given index
   virtual void roll(int index) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        if (index >= 0 && index < count) {
            diceValues[index] = rand() % 6 + 1; // Roll a new value from 1 to 6
        }
    }
    // Returns the total of all dice values
    virtual int total() const {
        int sum = 0;
        for (int i = 0; i < count; ++i) {
            sum += diceValues[i];
        }
        return sum;
    }
    // Returns a string representation of the dice values
    virtual std::string toString() const {
        std::ostringstream oss;
        oss << "{";
        for (int i = 0; i < count; ++i) {
            oss << diceValues[i];
            if (i < count - 1) {
                oss << ", ";
            }
        }
        oss << "}";
        return oss.str();
    }
    // Overloaded output operator
    friend std::ostream& operator<<(std::ostream& out, const Dice& dice) {
        out << dice.toString();
        return out;
    }
private:
    int* diceValues; // Array of dice values
    int count;       // Number of dice
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Inheritance : StackSorted

A class named ArrayStack has been written, which is an implementation of a stack of integers using an array as its internal representation.

```
class ArrayStack {
public:
  ArrayStack();               // construct empty stack
  ~ArrayStack();              // free memory

  virtual bool isEmpty() const;  // true if stack has no elements
  virtual int peek() const;      // return top element (error if empty)
  virtual int pop();             // remove/return top element (error if empty)
  virtual void push(int n);      // add to top of stack, resizing if needed

private:
  int* elements;              // array of stack data (index 0 = bottom)
  int size;                   // number of elements in stack
  int capacity;               // length of array
};
```

Define a new class called **StackSorted** that extends ArrayStack through inheritance. Your class represents a stack of integers that is always stored in sorted non-decreasing order, regardless of the order in which items are pushed onto the stack. Your class should provide the same member functions as the superclass. Your code must work with the existing ArrayStack as shown, unmodified. For example, if the following elements are added to an empty **StackSorted**:

42, 27, 39, 3, 55, 81, 11, 9, 0, 72

The following lines show the stack's state before and after adding each element, in bottom-to-top (left-to-right) order:

(You need to overload << operator for below output) "**ostream& operator <<(ostream& out, const StackSorted& stack);"**

```
{}
{42}
{27, 42}
{27, 39, 42}
{3, 27, 39, 42}
{3, 27, 39, 42, 55}
{3, 27, 39, 42, 55, 81}
{3, 11, 27, 39, 42, 55, 81}
{3, 9, 11, 27, 39, 42, 55, 81}
{0, 3, 9, 11, 27, 39, 42, 55, 81}
{0, 3, 9, 11, 27, 39, 42, 55, 72, 81}
```

You may create ArrayStack objects in your code if it is helpful to do so, but otherwise you should not create any auxiliary data structures (arrays, vectors, queues, maps, sets, strings, etc.) in your code.

Solution:

```
class StackSorted : public ArrayStack
{
public:
   StackSorted() : ArrayStack() {}
   void push(int n) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
        if (isEmpty()) {
            ArrayStack::push(n);
            return;
        }
        ArrayStack temp;
        while (!isEmpty() && peek() > n) {
            temp.push(pop());
        }
        ArrayStack::push(n);
        while (!temp.isEmpty()) {
            ArrayStack::push(temp.pop());
        }
    }
    friend ostream& operator<<(ostream& out, const StackSorted& stack);
};
ostream& operator<<(ostream& out, const StackSorted& stack) {
    ArrayStack temp;
    StackSorted* nonConstStack = const_cast<StackSorted*>(&stack);
    out << "{";
    while (!nonConstStack->isEmpty()) {
        temp.push(nonConstStack->pop());
    }
    bool first = true;
    while (!temp.isEmpty()) {
        int value = temp.pop();
        if (!first) {
            out << ", ";
        }
        out << value;
        nonConstStack->ArrayStack::push(value);
        first = false;
    }
    out << "}";
    return out;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Catch multiple exceptions

Complete the function cq1(int n) to throw below exceptions and catch them accordingly.
if n is negative, throw -1.if n is zero, throw "ZERO"if n is positive, throw 1.0

Implement necessary catch blocks accordingly as shown below.
if n is negative, print "NEGATIVE"if n is zero, print "ZERO"if n is positive, print "POSITIVE"

Solution:

```
#include <iostream>
#include <string>
using namespace std;
void cq1(int n)
{
    try {
        if (n < 0) {
            throw -1;
        } else if (n == 0) {
            throw "ZERO";
        } else if (n > 0) {
            throw 1.0;
        }
    }
    catch (int e) {
        if (e == -1) {
            cout << "NEGATIVE" << endl;
        }
    }
    catch (const char* msg) {
        cout << "ZERO" << endl;
    }
    catch (double e) {
        if (e == 1.0) {
            cout << "POSITIVE" << endl;
        }
    }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Password too short Exception

Write a program to accept user name and password from the user.
If the password has less than 6 characters then throw an exception as character 's'If the password does not contain a digit then throw an exception as character 'd'.
For handling exception implement necessary catch blocks and print the messages accordingly.

Sample Input1
Arun
abcd123
SampleOutput1
Correct
Sample Input2
Arun
abcdefgh
SampleOutput2
No digit!
Sample Input3
Arun
abc1
SampleOutput3
Too short!

Solution:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
   string username, password;
   cin >> username;
   cin >> password;
   try {
      if (password.length() < 6) {
         throw 's';
      }
      bool hasDigit = false;
      for (char c : password) {
         if (isdigit(c)) {
            hasDigit = true;
            break;
         }
      }
      if (!hasDigit) {
         throw 'd';
      }
      cout << "Correct" << endl;
   }
   catch (char exceptionType) {
      if (exceptionType == 's') {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            cout << "Too short!" << endl;
        }
        else if (exceptionType == 'd') {
            cout << "No digit!" << endl;
        }
    }
    return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Valid email address

Check an email-id for following exceptions
if id does not contain '@' **OR**if id does not contain '.' (dot) **OR**if no '.' (dot) appears after '@'
catch the exception and print "**Invalid**" or "**Valid**" accordingly.

Solution:

```
#include <iostream>
#include <string>
using namespace std;
void isvalidemail(string id)
{
   try {
      // Check if the ID contains '@'
      size_t atPos = id.find('@');
      if (atPos == string::npos) {
         throw 'a'; // No '@' found
      }
      // Check if the ID contains '.'
      size_t dotPos = id.find('.');
      if (dotPos == string::npos) {
         throw 'b'; // No '.' found
      }
      // Check if '.' appears after '@'
      if (dotPos <= atPos) {
         throw 'c'; // '.' does not appear after '@'
      }
      // If no exceptions, the email ID is valid
      cout << "Valid" << endl;
   }
   catch (char exceptionType) {
      if (exceptionType == 'a' || exceptionType == 'b' || exceptionType == 'c') {
         cout << "Invalid" << endl;
      }
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Valid index of array

Write a program to check whether the indexes given are valid index are not. Print "**Out of Bounds**" if any attempt is made to refer to an element whose index is beyond the array size , else print the element present at the index.

Input Format

The first line of input contains n , denoting the size of array.
The second line contains n spaced integers , denoting elements of array
The third line contains q, denoting number of queries
Next q lines contains an integer, index, which is to be tested.

Output Format

For each query print the value at index if the index is a valid index , else print "Out of Bounds".

Sample Input

```
10      // Size of array
1 2 3 4 5 6 7 8 9 10  // array elements
2 // No. Of Queries
4    // index to be retrieved
13    // index to be retrieved
```

Sample Output

```
5
Out of Bounds
```

Solution:

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   int n;
   cin >> n;
   vector<int> array(n);
   for (int i = 0; i < n; ++i) {
      cin >> array[i];
   }
   int q;
   cin >> q;
   for (int i = 0; i < q; ++i) {
      int index;
      cin >> index;
      if (index >= 0 && index < n) {
         cout << array[index] << endl;
      } else {
         cout << "Out of Bounds" << endl;
      }
   }
   return 0;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Swap two variables using templates

Write a function **Swap()** to swap two variables using templates, so that it can be used to swap any kind of variables.

Input Format

First Line of input contains 2 Integers seperated by a space
Second Line of Input contains 2 floating point number seperated by a space
Third Line of input contains 2 characters seperated by a space

Output Format

First 3 Lines contains the variables scanned without swapping
Next 3 Lines contains the variables scanned after swapping

Sample Input

```
1 2
1.1 2.2
a b
```

Sample Output

```
1 2
1.1 2.2
a b
2 1
2.2 1.1
b a
```

Solution:

```
#include <iostream>
using namespace std;
template <typename T>
void Swap(T &a, T &b) {
    T temp = a;
    a = b;
    b = temp;
}
void HandleSwap() {
    int int1, int2;
    cin >> int1 >> int2;
    float float1, float2;
    cin >> float1 >> float2;
    char char1, char2;
    cin >> char1 >> char2;
    cout << int1 << " " << int2 << endl;
    cout << float1 << " " << float2 << endl;
    cout << char1 << " " << char2 << endl;
    Swap(int1, int2);
    Swap(float1, float2);
    Swap(char1, char2);
    cout << int1 << " " << int2 << endl;
    cout << float1 << " " << float2 << endl;
    cout << char1 << " " << char2 << endl;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Calculator using templates

Write a class ***Calculate*** to add, subtract, multiply and divide two numbers using class templates.
Include methods below:
a constructor to initialize with two variables.a add() function to perform additiona sub() function to perform subtractiona mul() function to perform multiplicationa div() function to perform division

Solution:

```
#include <iostream>
using namespace std;
template <class T>
class Calculate {
private:
   T a, b;
public:
   Calculate(T x, T y) : a(x), b(y) {}
   T add() const {
      return a + b;
   }
   T sub() const {
      return a - b;
   }
   T mul() const {
      return a * b;
   }
   T div() const {
      if (b != 0) {
         return a / b;
      } else {
         throw runtime_error("Division by zero");
      }
   }
};
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Square of a number using templates

Write a function square as template to return the square of the argument.

Solution:

```
#include <iostream>
using namespace std;
template <typename T>
T square(T x) {
   return x * x;
}
void demonstrateSquare() {
   int intVal = 5;
   cout << "Square of " << intVal << " is " << square(intVal) << endl;
   double doubleVal = 3.14;
   cout << "Square of " << doubleVal << " is " << square(doubleVal) << endl;
   float floatVal = 2.5f;
   cout << "Square of " << floatVal << " is " << square(floatVal) << endl;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find minimum of a template array

Write a function ***minElement*** which takes an template array and and integer size of array as two arguments.
It must return the minimum of the array values, those can be int, float and strings as well).

Solution:

```
#include <iostream>
#include <string>
#include <limits>
using namespace std;
template <typename T>
T minElement(T arr[], int size) {
    if (size <= 0) {
        throw invalid_argument("Array size must be positive");
    }
    T minValue = arr[0];
    for (int i = 1; i < size; ++i) {
        if (arr[i] < minValue) {
            minValue = arr[i];
        }
    }
    return minValue;
}
void demonstrateMinElement() {
    int intArr[] = {3, 1, 4, 1, 5};
    int intSize = sizeof(intArr) / sizeof(intArr[0]);
    cout << "Minimum integer: " << minElement(intArr, intSize) << endl;
    float floatArr[] = {3.1f, 1.4f, 4.1f, 1.6f, 5.0f};
    int floatSize = sizeof(floatArr) / sizeof(floatArr[0]);
    cout << "Minimum float: " << minElement(floatArr, floatSize) << endl;
    string strArr[] = {"apple", "banana", "cherry", "date"};
    int strSize = sizeof(strArr) / sizeof(strArr[0]);
    cout << "Minimum string: " << minElement(strArr, strSize) << endl;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find cumulative sum of a vector

Write a function named **cumulative** that accepts a reference to a Vector of integers and modifies it so that each element contains the cumulative sum of the elements up through that index.
For example, if the vector passed contains **{1, 2, 3, 4, 5}**, your function should modify it to store **{1, 3, 6, 10, 15}**.

Input Format

The first line of input consists of an integer, N
The next N lines contains an integer.

Output Format

Print the vector elements after performing the operation seperated by a space.

Sample Input

5
1
2
3
4
5

Sample Output

1 3 6 10 15

Solution:

```
#include <iostream>
#include <vector>
void cumulative(std::vector<int>& v) {
   if (v.empty()) {
      return;
   }
   int currentSum = 0;
   for (size_t i = 0; i < v.size(); ++i) {
      currentSum += v[i];
      v[i] = currentSum;
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find mean of a vector

Write a function named ***mean()*** that accepts a reference to a vector of integers, and returns the arithmetic mean (average) of the integers in the vector as a double number.
For example, if the vector **v** contains **{4.5, 20.0, 1.5, 7.0}**, the call of ***mean(v)*** should return **8.25**.
If the vector is empty, return **0.0**.
Input Format
The first line of input consists of an integer, N
The next N lines contains a floating point number.
Output Format
Print the mean of the vector
Sample Input
4
4.0
20.0
1.5
7.0
Sample Output
8.25

Solution:

```
#include <iostream>
#include <vector>
double mean(const std::vector<double>& v) {
    if (v.empty()) {
        return 0.0;
    }
    double sum = 0.0;
    for (double num : v) {
        sum += num;
    }
    return sum / v.size();
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Count in range from a vector

Write a function named ***countInRange()*** that accepts three parameters:
a reference to a Vector of integers,a minimum anda maximum integer,
and returns the number of elements in the vector within that range inclusive.
For example, if the vector v contains {18, 1, 17, 4, 81, 9, 159, 8, 31, 30, 22}, the call of ***countInRange(v, 10, 30)*** should return 4.
If the vector is empty, return 0.

Input Format

The first line of input consists of an integer, N
The next N lines contains an Integer.
Then next 2 lines contain the minimum value and the maximum value of the range

Output Format

Print the count of elements present in the range

Sample Input

```
11
28
1
17
4
41
9
59
8
31
30
25
10
30
```

Sample Output

```
4
```

Solution:

```
#include <iostream>
#include <vector>
int countInRange(const std::vector<int>& v, int min, int max) {
    int count = 0;
    for (int num : v) {
        if (num >= min && num <= max) {
            ++count;
        }
    }
    return count;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Split the vector elements in two

Write a function named ***split()*** that accepts a reference to a Vector of integers, and modifies it to be twice as large, replacing every integer with a pair of integers, where each is half of the original.
If a number in the original vector is odd, then the first number in the new pair should be one higher than the second so that the sum equals the original number.
For example, if a variable named v refers to a vector storing the values {22, 45}, the call of **split(v)**; should change v to contain {11, 11, 23, 22}.
(The number 22 is splitted into the pair 11, 11, the number 45 is splitted into 23, 22.)
Input Format
The first line contains an integer N, denoting the size of vector
The second line contains N spaced integers
Output Format
Print the vector elements separated by space
Sample Input
2
22 45
Sample Output
11 11 23 22

Solution:

```
#include <iostream>
#include <vector>
void split(std::vector<int>& v) {
   std::vector<int> result;
   for (int num : v) {
      if (num % 2 == 0) {
         int half = num / 2;
         result.push_back(half);
         result.push_back(half);
      } else {
         int first = num / 2 + 1;
         int second = num / 2;
         result.push_back(first);
         result.push_back(second);
      }
   }
   v = result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Beautify the vector

Write a function named **beautify()** that accepts a reference to a vector of strings, and modifies the vector by placing a "#" element between elements, as well as at the start and end of the vector.
For example, if the vector v contains **{"Code", "Quotient"}**, the call of **beautify(v)** should modify it to store **{"#", "Code", "#", "Quotient", "#"}**.
Input Format
The first line of input consists of an integer, N
The next N lines contains a string.
Output Format
Print the vector after performing the operation
Sample Input
2
Code
Quotient
Sample Output
Code # Quotient

Solution:

```
#include <iostream>
#include <vector>
#include <string>
void beautify(std::vector<std::string>& v) {
    std::vector<std::string> result;
    result.push_back("#");
    for (const auto& str : v) {
        result.push_back(str);
        result.push_back("#");
    }
    v = result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Switch numbers in a vector

Write a function named ***switchPairs()*** that accepts a reference to a Vector of integers and it switches the order of values in a pairwise fashion.
It should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on.
For example, if the vector initially stores **{5, 14, 34, -13, 3, 8}**, then after calling **switchPairs()** it must be **{14, 5, -13, 34, 8, 3}**.
If there are an odd number of values in the list, the final element is not moved. For example, if the original vector had been **{5, 14, 34, -13, 3, 8, 7}**, then the result would be **{14, 5, -13, 34, 8, 3, 7}**.

Input Format

The first line of input consists of an integer, N
The next line contains N integers seperated by a space.

Output Format

Print the vector after performing the operation

Sample Input - 1

6
5 14 34 -13 3 8

Sample Output - 1

14 5 -13 34 8 3

Sample Input - 2

7
5 14 34 -13 3 8 7

Sample Output - 2

14 5 -13 34 8 3 7

Solution:

```
#include <iostream>
#include <vector>
void switchPairs(std::vector<int>& v) {
   for (size_t i = 0; i + 1 < v.size(); i += 2) {
      std::swap(v[i], v[i + 1]);
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Remove an element from vector of string

Write a function named ***removeAll()*** that accepts a reference to a vector of strings along with a string to be removed , and modifies the vector to remove all occurrences of that string from vector.
For example, if the vector v contains **{"Code", "Quotient", "Coding", "Quotient", "Coding", "Coding", "Code"}**,
then call of **removeAll(v, "Coding");** should modify it to store **{"Code", "Quotient", "Quotient", "Code"}**.
Input Format
The first line of input contains an integer N, denoting the size of vector
The next line contains N space separated strings.
The next line contains the string to be removed.
Output Format
Print the elements of vector separated by a space

Sample Input
7
Code Quotient Coding Quotient Coding Coding Code
Coding
Sample Output
Code Quotient Quotient Code

Solution:

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
void removeAll(std::vector<std::string>& v, const std::string& toRemove) {
    auto newEnd = std::remove(v.begin(), v.end(), toRemove);
    v.erase(newEnd, v.end());
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Remove consecutive duplicates from a vector

Write a function named ***removeConsecutiveDups()*** that accepts as a parameter a reference to a Vector of integers,
and modifies it by removing any consecutive duplicates.
For example, if a vector named v stores {2, 5, 5, 7, 6, 6, 9}, the call of removeConsecutiveDups(v); should modify it to store {2, 5, 7, 6, 9}.

Solution:

```
#include <vector>
void removeConsecutiveDups(std::vector<int>& v) {
   if (v.empty()) return; // If the vector is empty, there's nothing to do
   // Create a vector to store the result
   std::vector<int> result;
   // Initialize with the first element
   result.push_back(v[0]);
   // Iterate through the vector starting from the second element
   for (size_t i = 1; i < v.size(); ++i) {
      if (v[i] != v[i - 1]) {
         // Add to result if not equal to the previous element
         result.push_back(v[i]);
      }
   }
   // Update the original vector to match the result
   v = result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Remove odd length strings from a vector of strings

Write a function named ***removeOdd()*** that accepts as a parameter
a reference to a Vector of strings,
and modifies the vector to remove all occurrences of those strings whose length is odd.
For example, if the vector v contains {"Code", "Quotient", "Get", "Better", "@", "Coding"},
then call of **removeOdd(v);** should modify it to store {"Code", "Quotient", "Better",
"Coding"}.

Solution:

```
#include <vector>
#include <string>
void removeOdd(std::vector<std::string>& v) {
    // Create a vector to store the result
    std::vector<std::string> result;
    // Iterate through the vector
    for (const auto& str : v) {
        // Check if the length of the string is even
        if (str.length() % 2 == 0) {
            result.push_back(str);
        }
    }
    // Update the original vector to match the result
    v = result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Remove large Pairs from a vector of integers

Write a function named **removeLargePairs()** that accepts as a parameter
a reference to a Vector of integers,
and removes any adjacent pair of integers in the list if the left element of the pair is larger than the right element of the pair. For example, suppose a vector named v stores the following element values:
{1, 1, 2, 2, 4, 3, 6, 5, 3, 7, 8, 5}
We can think of this list as a sequence of pairs:
{{1, 1}, {2, 2}, {4, 3}, {6, 5}, {3, 7}, {8, 5}}
The pairs {4, 3}, {6, 5} and {8, 5} are "large" because the left element is larger than the right one, so these pairs should be removed. So the call of **removeLargePairs**(v); would change the vector to store:
{1, 1, 2, 2, 3, 7}

If the vector has an odd length, the last element is not part of a pair and is left untouched.If an empty vector is passed in, the vector should still be empty at the end of the call.Do not use any other arrays, vectors, or other data structures to help solve this problem, though you can create as many simple variables as you like.

Solution:

```
#include <vector>
void removeLargePairs(std::vector<int>& v) {
    // Create a vector to store the result
    std::vector<int> result;
    // Iterate through the vector by pairs
    size_t i = 0;
    while (i + 1 < v.size()) {
        if (v[i] <= v[i + 1]) {
            // Add the valid pair to the result vector
            result.push_back(v[i]);
            result.push_back(v[i + 1]);
        }
        // Move to the next pair
        i += 2;
    }
    // If there's an odd number of elements, add the last element
    if (v.size() % 2 != 0) {
        result.push_back(v.back());
    }
    // Update the original vector to match the result
    v = result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Intersection of two vectors

Write a function named **intersection()** that accepts
references to two sorted Vector of integers as parameters
and returns a new vector that contains only the elements that are found in both vectors. For example, if vector v1 and v2 store:
v1 = {1, 2, 3, 4, 5, 6}
v2 = {2, 4, 6, 8}
Then the call of **intersection(v1, v2)** returns the vector: {2, 4, 6}

Solution:

```
#include <vector>
std::vector<int> intersection(const std::vector<int>& v1, const std::vector<int>& v2) {
    std::vector<int> result;
    size_t i = 0, j = 0;
    // Traverse both vectors
    while (i < v1.size() && j < v2.size()) {
        if (v1[i] < v2[j]) {
            ++i; // Move forward in v1
        } else if (v1[i] > v2[j]) {
            ++j; // Move forward in v2
        } else {
            // If both elements are equal, add to result and move both pointers
            result.push_back(v1[i]);
            ++i;
            ++j;
        }
    }
    return result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find number is repfigit number or not

Write a function named **isrepfigitNumber()** that accepts an integer and returns true if that number is a "repfigit number" (**rep**etitive **F**ibonacci-like d**igit**).
A "repfigit number" is defined as any n-digit integer that appears in the sequence that starts off with the number's n digits and then continues such that each subsequent number is the sum of the preceding n. (This is not unlike the classic Fibonacci sequence.)
All one-digit numbers are trivially repfigit numbers.
As an example, consider the 3-digit number *N* = 197. The sequence goes like this:
1, **9**, **7**, 17, 33, 57, 107, 197, 361, ...
Because 197 appears in the sequence, 197 is seen to be indeed a repfigit number.
The sequence starts out 1, 9, 7, because those are the digits making up 197. Each number after that is the sum of the three numbers that precede it (three, because 197 has three digits). So the forth number is the sum of 1+9+7=17. The fifth number is 9+7+17=33 and so on, until we eventually get back to 197, which makes 197 a repfigit number.
You may use a single Vector or LinkedList as auxiliary storage.
Your function should not loop infinitely; if you become sure that the number is not a repfigit number, stop searching and immediately return false.

Solution:

```
#include <vector>
#include <deque>
#include <algorithm>
bool isrepfigitNumber(int n) {
    // Handle single-digit numbers directly
    if (n >= 0 && n < 10) return true;
    // Convert number to its digit representation
    std::vector<int> digits;
    int temp = n;
    while (temp > 0) {
        digits.push_back(temp % 10);
        temp /= 10;
    }
    std::reverse(digits.begin(), digits.end());
    int length = digits.size();
    std::deque<int> sequence(digits.begin(), digits.end());
    // Generate the sequence
    while (true) {
        int sum = 0;
        for (int i = 0; i < length; ++i) {
            sum += sequence[i];
        }
        // Check if the new number matches the original number
        if (sum == n) return true;
        // If the number exceeds `n` and it has wrapped around, return false
        if (sum > n && sum > sequence.back()) return false;
        // Remove the oldest number and add the new one to the sequence
        sequence.pop_front();
        sequence.push_back(sum);
        // Avoid infinite loop by limiting the length of the sequence
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            if (sequence.size() > 2 * length) return false;
        }
    }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Duplicate the Queue elements

Write a function named **doubleQueue()** that accepts
a reference to a queue of integers as a parameter
and replaces every element with two copies of itself.
For example, if a queue named q stores {11, 12, 13}, the call of **doubleQueue**(q); should change it to store {11, 11, 12, 12, 13, 13}.
Constraints: Do not use any auxiliary collections as storage.

Solution:

```
#include <queue>
void doubleQueue(std::queue<int>& q)
{
   int size = q.size();
   // Step 1: Process each element and duplicate it
   for (int i = 0; i < size; ++i) {
      int elem = q.front();
      q.pop();
      q.push(elem);  // Re-insert the element once
      q.push(elem);  // Re-insert the element a second time
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Get the mirror of the queue

Write a function named **mirrorQueue()** that accepts
a reference to a queue of strings as a parameter
and appends the queue's contents to itself in reverse order.
For example, if a queue named q stores {"Code", "Quotient"}, the call of **mirrorQueue(q);**
should change it to store {"Code", "Quotient", "Quotient", "Code"}.

Solution:

```
#include <queue>
#include <stack>
#include <string>
#include <iostream>
void mirrorQueue(std::queue<std::string>& q)
{
   std::stack<std::string> s;
   int size = q.size();
   for (int i = 0; i < size; ++i) {
      std::string elem = q.front();
      q.pop();
      s.push(elem);
      q.push(elem);
   }
   while (!s.empty()) {
      q.push(s.top());
      s.pop();
   }
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Check for balanced parentheses in string

Complete the function **balancedString()** that accepts a string of source code and check whether the braces/parentheses are balanced.
Every (or { must be closed by a } or) in the opposite order. Return the index at which an imbalance occurs, or -1 if the string is balanced. If any (or { are never closed, return the string's length.
Here are some example calls:
balancedString("if(a(4)>9){foo(a(2));}") // returns -1 because balanced
balancedString("for(i=0;i<a(3};i++){foo{);)") // returns 13 because } out of order
balancedString("while(true)foo();}{()") // returns 17 because } doesn't match any {
balancedString("if(x){") // returns 6 because { is never closed
Input Format:
Input consists of a single line that contains the string of source code.
Output Format:
Return the index at which the imbalance occurs, if the string is balanced return -1
Sample Input
if(a(4)>9){foo(a(2));}
Sample Output
-1
Explanation
The given string is balanced, hence the output is -1þÿ
Sample Input
for(i=0;i<a(3};i++){foo{);)
Sample Output
13
Explanation
The given string is imbalanced at } on index 13þÿ

Solution:

```
#include <string>
#include <stack>
int balancedString(const std::string& s) {
    std::stack<char> st;
    for (int i = 0; i < s.size(); ++i) {
        char c = s[i];
        if (c == '(' || c == '{') {
            st.push(c);
        } else if (c == ')' || c == '}') {
            if (st.empty()) {
                return i; // Closing bracket without matching opening
            }
            char top = st.top();
            st.pop();
            if ((c == ')' && top != '(') || (c == '}' && top != '{')) {
                return i; // Mismatched closing bracket
            }
        }
    }
    if (!st.empty()) {
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            return s.size(); // Unmatched opening brackets remaining
        }
        return -1; // Balanced
    }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Flip the odd elements of queue

Write a function named **flipHalfQueue()** that
Takes an queue of integers as parameter
and reverses the order of half of the elements of a Queue of integers,
Your function should reverse the order of all the elements in odd-numbered positions (position 1, 3, 5, etc.) assuming that the first value in the queue has position 0.
For example, if the queue originally stores this sequence of numbers when the function is called:
index: 0 1 2 3 4 5 6 7
front:{5, 7, 6, 2, 9, 18, 11, 15} back
Then it should store the following values after the function finishes executing:
index: 0 1 2 3 4 5 6 7
front:{5, 15, 6, 18, 9, 2, 11, 7} back
Notice that numbers in even positions (positions 0, 2, 4, 6) have not moved. That sub-sequence of numbers is still: (5, 6, 9, 11).
But notice that the numbers in odd positions (positions 1, 3, 5, 7) are now in reverse order relative to the original. In other words, the original sub-sequence: (7, 2, 18, 15) - has become: (15, 18, 2, 7).
Constraints: You may use a single stack as auxiliary storage.

Solution:

```
#include <iostream>
#include <queue>
#include <stack>
void flipHalfQueue(std::queue<int>& q) {
    std::stack<int> s;
    std::queue<int> tempQueue;
    // Transfer elements to a temporary queue while processing odd positions
    int index = 0;
    while (!q.empty()) {
        int current = q.front();
        q.pop();
        if (index % 2 == 1) { // Odd index
            s.push(current);
        }
        tempQueue.push(current);
        index++;
    }
    // Reinsert elements back into the original queue
    index = 0;
    while (!tempQueue.empty()) {
        int current = tempQueue.front();
        tempQueue.pop();
        if (index % 2 == 1) { // Odd index
            q.push(s.top());
            s.pop();
        } else {
            q.push(current);
        }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
            index++;
        }
    }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Number is a Happy number or not

Write a function named **isHappy()** that returns whether a given integer is "happy" or not.
An integer is "happy" if repeatedly summing the squares of its digits eventually leads to the number 1.
For example, 139 is happy because:
12 + 32 + 92 = 9192 + 12 = 8282 + 22 = 6862 + 82 = 10012 + 02 + 02 = 1
By contrast, 4 is not happy because:
42 = 1612 + 62 = 3732 + 72 = 5852 + 82 = 8982 + 92 = 14512 + 42 + 52 = 4242 + 22 = 2022 + 02 = 4...

Note: Don't stuck in an infinite loop.

Solution:

```
#include <iostream>
#include <unordered_set>
int sumOfSquares(int n) {
    int sum = 0;
    while (n > 0) {
        int digit = n % 10;
        sum += digit * digit;
        n /= 10;
    }
    return sum;
}
bool isHappy(int n) {
    std::unordered_set<int> seen;
    while (n != 1 && seen.find(n) == seen.end()) {
        seen.insert(n);
        n = sumOfSquares(n);
    }
    return n == 1;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Find the numbers appearing thrice

Write a function named **thrice()** that accepts as a parameter
a reference to a vector of integers
and returns a set containing all the numbers in the vector that appear exactly thrice.
For example, if a vector variable v stores {1, 1, 1, 4, 13, 17, -12, 33, 17, 82, 17, 61}, the call of **thrice(v)** should return the set {1, 17}.

Solution:

```
#include <vector>
#include <unordered_map>
#include <set>
std::set<int> thrice(std::vector<int>& v) {
    std::unordered_map<int, int> countMap;
    std::set<int> result;
    for (int num : v) {
        countMap[num]++;
    }
    for (const auto& entry : countMap) {
        if (entry.second == 3) {
            result.insert(entry.first);
        }
    }
    return result;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: Remove duplicates from a vector

Write a function named removeDuplicates() that accepts as a parameter
a reference to a Vector of integers,
and modifies it by removing any duplicates.
Note that the elements of the vector are not in any particular order, so the duplicates might not occur consecutively.
You should retain the original relative order of the elements.
Use a Set as auxiliary storage to help you solve this problem.
For example, if a vector named v stores {24, 10, 12, 19, 24, 17, 12, 10, 10, 19, 26, 26}, the call of **removeDuplicates(v);** should modify it to store {24, 10, 12, 19, 17, 26}.

Solution:

```
#include <vector>
#include <set>
void removeDuplicates(std::vector<int>& v) {
   std::set<int> seen;
   std::vector<int> uniqueElements;
   for (int num : v) {
      // Check if the element is not in the set
      if (seen.find(num) == seen.end()) {
         // Add to the set and uniqueElements vector
         seen.insert(num);
         uniqueElements.push_back(num);
      }
   }
   // Replace the original vector with the unique elements
   v = uniqueElements;
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Aim: C++ Problem Solution

Write a C++ application to create Patient class with id,age,name bGroup attribute and its constructor and getter setter
property functions create an array of pointer to create dynamic input of patient and store their data into this collection
and show the list of patient having A+ bGroup as A and otherwise as @ and at end show the count of patient A+
Hint: Array of Pointer, Class Object

- **Create Patient class with id, age name bGroup**
- **Getter and setter function**
- **Main with array of pointers to take dynamic input and create object entries**
- **Output @ for bgroup not found and A for A+ found and next is count of A+ blood group**

Input:
3
1 12 sai B+
2 13 ram O+
3 14 raja AB
Output:
@@@
0

Input:
10
Output:
Input Constraint Error range.

Input:
3
1 11 ram A+
2 12 sita A+
3 13 radhe O+

Output:
AA@
2

ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

Solution:

```
private:
    int id;
    int age;
    string name;
    string bGroup;
public:
    // Constructor
    Patient(int id, int age, string name, string bGroup) {
        this->id = id;
        this->age = age;
        this->name = name;
        this->bGroup = bGroup;
    }
    // Getter and Setter functions
    int getId() { return id; }
    void setId(int id) { this->id = id; }
    int getAge() { return age; }
    void setAge(int age) { this->age = age; }
    string getName() { return name; }
    void setName(string name) { this->name = name; }
    string getBGroup() { return bGroup; }
    void setBGroup(string bGroup) { this->bGroup = bGroup; }
};
int main() {
    int n;
    cin >> n;
    if (n <= 0 || n > 3) { // Adjusted constraint to match the expected output
        cout << "Input Constraint Error range." << endl;
        return 0;
    }
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in

```
Patient* patients[n];
for (int i = 0; i < n; i++) {
    int id, age;
    string name, bGroup;
    cin >> id >> age >> name >> bGroup;
    patients[i] = new Patient(id, age, name, bGroup);
}
int count = 0;
for (int i = 0; i < n; i++) {
    if (patients[i]->getBGroup() == "A+") {
        cout << "A";
        count++;
    } else {
        cout << "@";
    }
}
// Free allocated memory
for (int i = 0; i < n; i++) {
    delete patients[i];
}
```


ANUJ KUMAR
anuj1699.be23@chitkara.edu.in