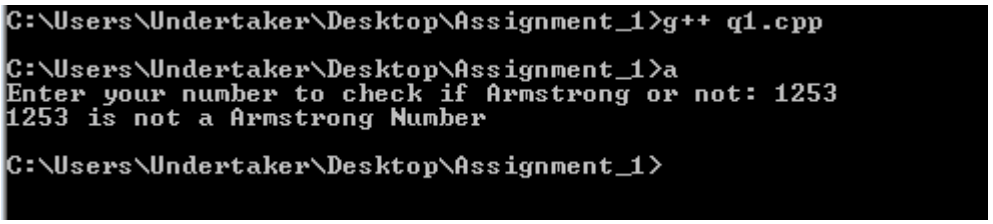
Name : SAHIB SINGH Assignment 1 :9-jun-2021

Q.1 Given a number x, determine whether the given number is Armstrong number or not. A positive integer of **n digits** is called an Armstrong number of **order n** (order is number of digits) if.

abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ...

```
Input : 1253
Output : No
1253 is not a Armstrong Number
```

Screenshot:



Source code:

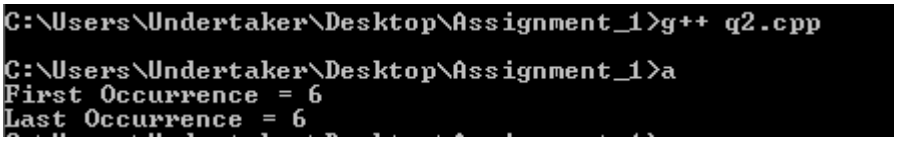
```cpp
#include<bits/stdc++.h>
using namespace std;

#include<bits/stdc++.h>
using namespace std;

/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
    if( y == 0)
        return 1;
    if (y%2 == 0)
        return power(x, y/2)*power(x, y/2);
    return x*power(x, y/2)*power(x, y/2);
}

/* Function to calculate order of the number */
int order(int x)
{
    int n = 0;
    while (x)
    {
        n++;
        x = x/10;
    }
    return n;
```

```
}

// Function to check whether the given number is
// Armstrong number or not
bool isArmstrong(int x)
{
    // Calling order function
    int n = order(x);
    int temp = x, sum = 0;
    while (temp)
    {
        int r = temp%10;
        sum += power(r, n);
        temp = temp/10;
    }

    // If satisfies Armstrong condition
    return (sum == x);
}


int main()
{
    int x;
    cout<<"Enter your number to check if Armstrong or not: ";
    cin>>x;
    if(isArmstrong(x)==0)
    {
        cout<<x<<" is not a Armstrong Number"<<endl;

    }
    else if (isArmstrong(x)==1)
    {
        cout<<x<<" is a Armstrong Number"<<endl;

    }
    return 0;
}
```

Q.2 Given a sorted array with possibly duplicate elements, the task is to find indexes of first and last occurrences of an element x in the given array.

```
Input : arr[] = {1, 3, 5, 5, 5, 5, 7, 123, 125 }
        x = 7
Output : First Occurrence = 6
         Last Occurrence = 6
```

Screenshot:



Source Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

// Function for finding first and last occurrence
// of an elements
void findFirstAndLast(int arr[], int n, int x)
{
    int first = -1, last = -1;
    for (int i = 0; i < n; i++) {
        if (x != arr[i])
            continue;
        if (first == -1)
            first = i;
        last = i;
    }
    if (first != -1)
        cout << "First Occurrence = " << first
             << "\nLast Occurrence = " << last;
    else
        cout << "Not Found";
}


int main()
{
    int arr[] = { 1, 3, 5, 5, 5, 5, 7, 123, 125  };
    int n = sizeof(arr) / sizeof(int);
    int x = 7;
    findFirstAndLast(arr, n, x);
    return 0;
}
```

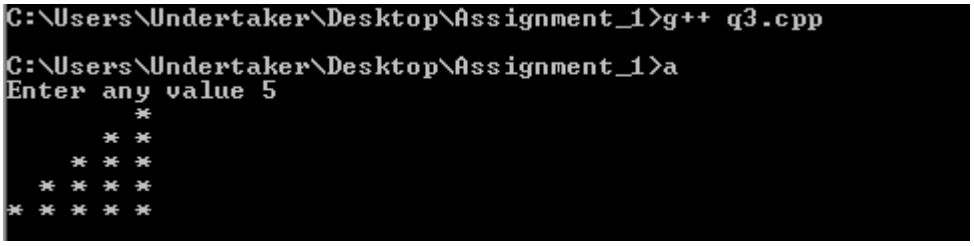
Q.3 **1. You are given a number n.**
2. You've to create a pattern of * and separated by tab as shown in output format.

Input : 5

Output:

```
                        *
                *       *
        *       *       *
*       *       *       *
```

Screenshot:



Source Code:

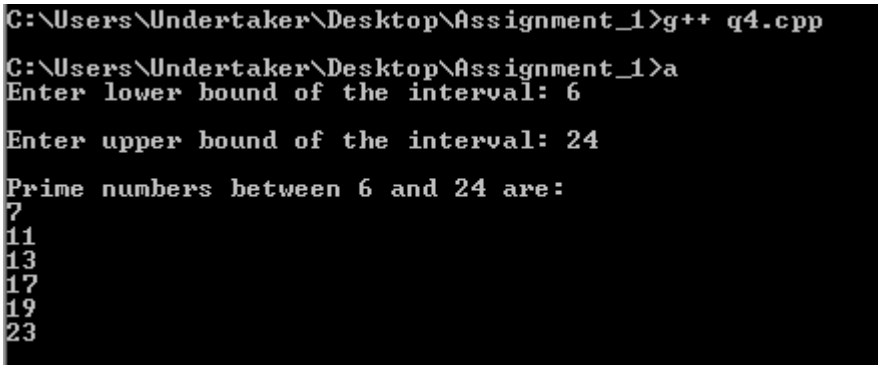
```cpp
#include <bits/stdc++.h>
using namespace std;

// Function to demonstrate printing pattern
void piramid(int n)
{
    // number of spaces
    int k = 2 * n - 2;

    // Outer loop to handle number of rows

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

        // Inner loop to handle number spaces
        // values changing acc. to requirement
        for (int j = 0; j < k; j++)
            cout << " ";

        // Decrementing k after each loop
        k = k - 2;

        // Inner loop to handle number of columns
        // values changing acc. to outer loop
        for (int j = 0; j <= i; j++) {
            // Printing stars
            cout << "* ";
```

```
        }

        // Ending line after each row
        cout << endl;
    }
}

// Driver Code
int main()
{
    int n;
    cout<<"Enter any value ";
    cin>>n;

      // Function Call
    piramid(n);
    return 0;
}
```

Q.4 **1. You've to print all prime numbers between a range.**
2. Take as input "low", the lower limit of range.
3. Take as input "high", the higher limit of range.
4. For the range print all the primes numbers between low and high (both included).

Input: low : 6
High: 24

Screen Shot:



Source Code:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // Declare the variables
    int a, b, i, j, flag;

    // Ask user to enter lower value of interval
    cout << "Enter lower bound of the interval: ";
    cin >> a; // Take input

    // Ask user to enter upper value of interval
    cout << "\nEnter upper bound of the interval: ";
    cin >> b; // Take input

    // Print display message
    cout << "\nPrime numbers between "
         << a << " and " << b << " are: "<<endl;

    // Traverse each number in the interval
    // with the help of for loop
    for (i = a; i <= b; i++) {
        // Skip 0 and 1 as they are
        // neither prime nor composite
        if (i == 1 || i == 0)
```

```
            continue;

        // flag variable to tell
        // if i is prime or not
        flag = 1;

        for (j = 2; j <= i / 2; ++j) {
            if (i % j == 0) {
                flag = 0;
                break;
            }
        }

        // flag = 1 means i is prime
        // and flag = 0 means i is not prime
        if (flag == 1)
            cout << i << " "<<endl;

    }

    return 0;
}
```

Q.5 1. You are given a string that contains only lowercase and uppercase alphabets.
2. You have to toggle the case of every character of the given string.

Input : ProGraMMer
Output: pROgRAmmER

Screenshot:

Source Code:

```
#include<bits/stdc++.h>
using namespace std;

void toggleChars(char str[])
{
    for (int i=0; str[i]!='\0'; i++)
    {
        if (str[i]>='A' && str[i]<='Z')
            str[i] = str[i] + 'a' - 'A';
        else if (str[i]>='a' && str[i]<='z')
            str[i] = str[i] + 'A' - 'a';
    }
}


int main()
{
    char str[10];
    cout<<"Type word that you want to  in toggle:: ";
    cin>>str;
    toggleChars(str);
    cout << "String after toggle = ";
    cout << str << endl;
    return 0;
}
```

**********OVER**********