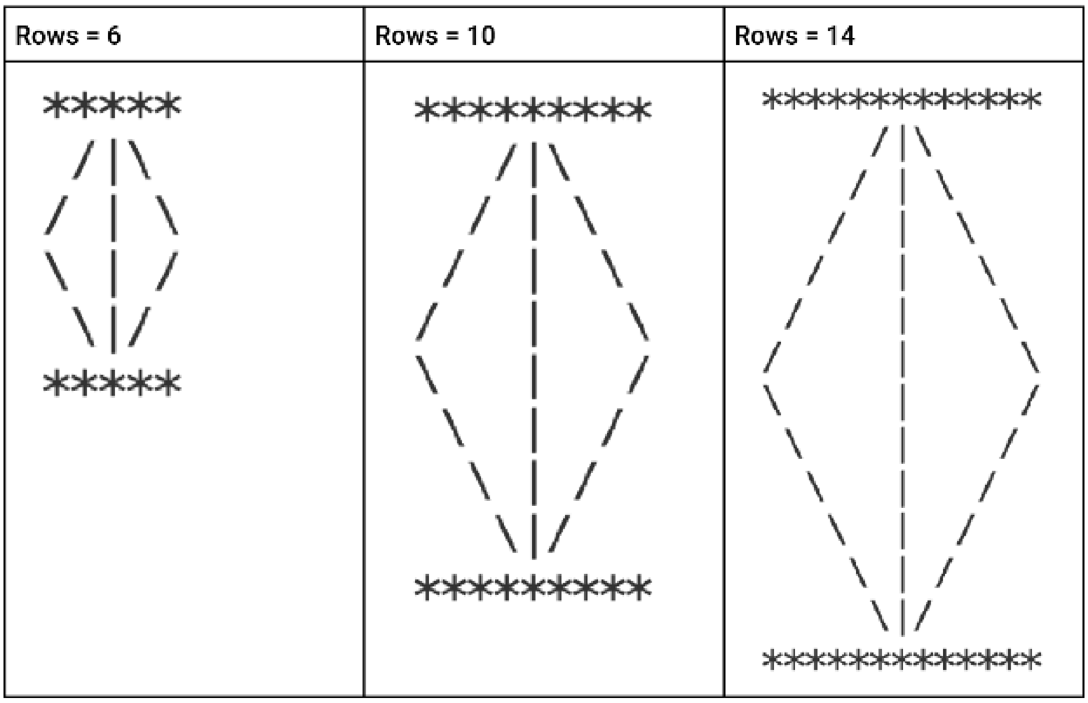
CS PF Lab

Mid 2023-2 Q1

Write a C++ program to draw the following pattern using loop(s). You have to input the number of rows. The number of rows will be an even number and greater than equal to four (4,6,8,10,12,14...). There is no need for input validation. 10 Bonus Marks for implementing in a single loop.



Rows = 6	Rows = 10	Rows = 14
see below	see below	see below

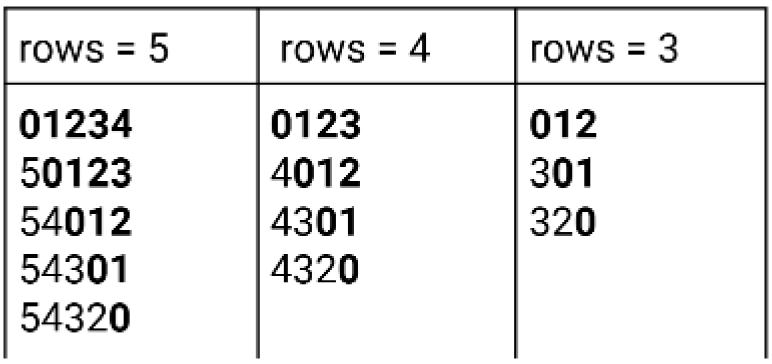
Rows = 6
```
*****
 /|\
/ | \
\ | /
 \|/
*****
```

Rows = 10
```
*********
   /|\
  / | \
 /  |  \
/   |   \
\   |   /
 \  |  /
  \ | /
   \|/
*********
```

Rows = 14
```
*************
     /|\
    / | \
   /  |  \
  /   |   \
 /    |    \
/     |     \
\     |     /
 \    |    /
  \   |   /
   \  |  /
    \ | /
     \|/
*************
```

Mid 2023-2 Q2

Write a C++ program to draw the following pattern using loop(s). You can't just print it with a cout statement. You have to input the number of rows. The number of rows will be in range [3, 4 & 5] only. But your program must be generic. There is no need for input validation. 10 Bonus Marks for implementing in a single loop.



rows = 5	rows = 4	rows = 3
01234	0123	012
50123	4012	301
54012	4301	320
54301	4320	
54320		

Final 2023 Q4

Develop a C++ program to convert a given floating-point number representing a currency value in rupees into its English phrase representation. Your program should handle values up to one lac (100,000). Following are the guidelines for implementing the currency conversion in C++:

- ➔ Accept a non-negative floating-point number, with at most two decimal places, as input. If more than 2 decimal places then round off to two decimal places.
- ➔ Output the English phrase representation of the provided currency value in rupees.
    - ◆ For instance, represent 12345 as "twelve thousand three hundred forty-five Rupees."
- ➔ Account for decimal values up to two decimal places.
    - ◆ For example, represent 12345.67 as "twelve thousand three hundred forty-five Rupees and sixty-seven Paisa."

DS PF Lab

Final 2023

Q1A

1. Prompt the user to enter a string s.
2. Copy this input string into another string t.
3. Append randomly generated characters to t to ensure its length is one character longer than s.
    a. Utilize ASCII values to generate these additional characters.

Example:
input : abcd
output: abcde
explanation: e is the newly added character

Q1B

Write a c++ program that determines the length of the longest substring without repeating characters within a given string.
Example 1:
Input: "abcabcbb"

Output: 3
Explanation: The longest substring without repeating characters is "abc", with a length of 3.
Example 2:
Input: " "
Output: 1
Explanation: The only character, " ", is a substring without repeating characters, with a length of 1.

Q2A

Given a 3x3 matrix in C++ (a 2D array), generate a sequence by reading its elements row by row, starting from the top-left corner and ending at the bottom-right corner. // do not hardcode ittt
Input matrix:
1 2 3
4 5 6
7 8 9
Output sequence: 124753689

Q2B

Create a C++ program that checks whether a given matrix is a Toeplitz matrix.
A Toeplitz matrix is defined as a diagonal-constant matrix, where all elements along a diagonal have the same value.
Input Matrix:
1 2 3 4
5 1 2 3
9 5 1 2
Output: Matrix is a Toeplitz

Q3A

Write a C++ program that swaps two numbers without using a third variable. You should utilize pointers and create a function named swap to perform the swapping operation. After swapping, display the updated numbers in the main function.

Q3B

Implement an Encrypt(int*arr, int size) function that takes an integer array and its size as parameters for encryption and a Decrypt(int*arr, int size) function for decryption. // experiment with multiple encryption formulae

Note: use pointers and functions

Sample:

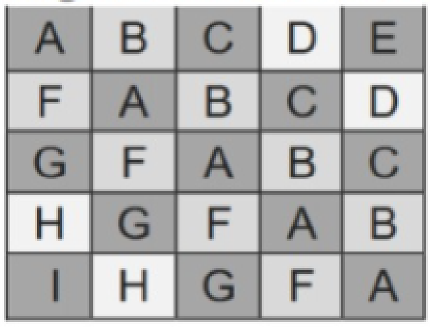
```
arr[5]={10,20,30,40,50}
encrypted array={80,160,240,320,400}
decrypted array={10,20,30,40,50}
```

Matrices

Diagonal Constant [30 Marks]

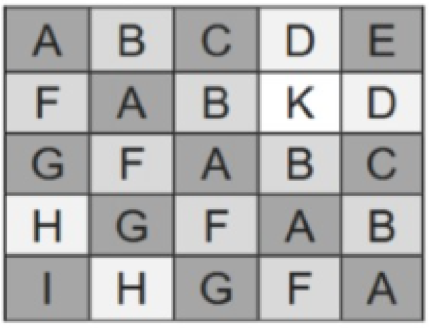
A diagonal-constant matrix is a square matrix in which each descending diagonal from left to right is constant with the same value.

A diagonal-constant Matrix



A	B	C	D	E
F	A	B	C	D
G	F	A	B	C
H	G	F	A	B
I	H	G	F	A

Not a diagonal-constant Matrix



A	B	C	D	E
F	A	B	K	D
G	F	A	B	C
H	G	F	A	B
I	H	G	F	A

Write a C++ function that takes, as parameters a 2D matrix and its dimensions and determines, whether it is a diagonal-constant matrix or not by returning true or false.

Dry Run 1.

```
int main() {
     int a;
     int b = 1;
     int x[5] = { 1, 2, 3, 4, 5 };
     a = 5 * 4 + x[--b] - (9 / b);
     cout << a;
}
```

Sum of Border Values of Matrix

Write a C program that calculates the sum of the elements along the border of a given square matrix of size n x n.
The program should:

- Take an input from the user to create a 2D array/matrix.

    - Calculate the sum of the elements along the border (Including corners).
    - Display the calculated sum to the user.

Input: 1 2 3 4 5 6 7 8 9

Process: (1+2+3)+(1+4+7)+(7+8+9) + (3+6+9)

Expected Output: Sum: 60

Jagged Array

You are designing a storage manager for a program that maintains several dynamic arrays of different sizes.

1. Initially, create a dynamic array of size 3, which points to other dynamic arrays whose size is provided by the user.
2. After filling the first three arrays, the user may choose to add more.
3. Each time the user decides to add another array, the root array must grow by 1.

Continue asking the user whether they want to create another array, and expand the root array each time the limit is reached.

Loops (and Recursion)

FizzBuzz from 1 to n

Write a program which iterates the integers from 1 to input ‘n’.
For multiples of three print "Fizz" instead of the number and for multiples of five print "Buzz".
For numbers which are multiples of both three and five print "FizzBuzz".
Sample Output:

```
fizzbuzz
1
2
fizz
4
buzz
```

Binary to Decimal Conversion

Write a program which accepts a sequence of separated n-digit binary numbers as its input and print the numbers’ decimal equivalent.

Sample Data: 0100
Length: 4
Expected Output: 8

Even-Digit Numbers Between 100–400

Write a program to find numbers between 100 and 400 (both included) where each digit of a number is an even number. The numbers obtained should be printed in a comma-separated sequence.

Median of Three Values

Write a program to find the median of n values.

Sum and Average Until Zero

Write a program to calculate the sum and average of n integer numbers (input from user). Input 0 to finish.

Palindrome Check

Input n, check if n is a palindrome

Harshad Number Checker

Write a program that determines whether an input number is a Harshad number (for number base 10).
A Harshad number “is an integer that is divisible by the sum of its digits” (Wikipedia).

```
 Example:
 81 → 8 + 1 = 9 → 81/9 = 9 → Harshad!
```

Factorial of an Input Number

Given an input number, print its factorial.

Divisible by 7 and Multiple of 5 Finder

Write a program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and n (both included).

Guess the Number Game

Write a program to guess a number between 1 to 9.
User is prompted to enter a guess. If the user guesses wrong then the prompt appears again until the guess is correct; on successful guess user will get a "Well guessed!" message, and the program will exit.

Count Odd and Even Until -1

Write a program to count the number of even and odd numbers from a series of numbers. Program should accept input until the user enters -1; it should then print the number of odds and evens.

Print Numbers Except Forbidden Multiples

Write a program that prints all the numbers from 0 to input ‘n’ except 3, 6, 12, 14 multiples.

Recursive Fibonacci up to n

!! IMP IN RECURSION

Write a program to get the Fibonacci series between 0 to input ‘n’.

```
 Expected Output:
 1 1 2 3 5 8 13 21 34
```

Star Patterns 1. Simple Triangles

Write a program to construct the following patterns of size ‘n’, following are for n=5:

a.

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

b.

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

c.

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

d.

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

e.

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

f.

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

Star Patterns 2. Lines with Gradients

In a grid, using spaces and stars, draw lines of slope(m):

- ½
- 2
- 1
- 0
- infinity

Star Patterns 3. Circle

Draw a circle of radius ‘n’

Star Patterns 4. Triangle

For n=4:

```
    *

   * *

  * * *

 * * * *
```

Star Patterns 5. Letters

Print the following letters using a. Nested loop, b. Single loop

1. Z
2. I
3. A
4. H
5. L
6. S
7. T
8. U
    a. sharp lower half
    b. Curved lower half
9. V
10. X
11. C (sharp)
12. O
    a. Square
    b. curved
13. 3
14. E
15. F
16. 6
17. 7
18. 8
19. 9

Star Patterns 6.

Make the following for size n(n=length of corner lines)

N = 7

```
......*..*......
.....*....*.....
....*......*....
```

```
...*........*...

..*..........*..

.*............*.

*..............*

................

................

*..............*

.*............*.

..*..........*..

...*........*...

....*......*....

.....*....*.....

......*..*......
```

Star X

Write a program to print alphabet pattern 'X' of height ‘n’.
 Expected Output for n = 5 :

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

Star Diamond

Generate the following diamond for size n, following is n=4:

```
   *

  * *

 * * *

* * * *

 * * *

  * *

   *
```

Arrow Pattern

Generate an arrow of size n, following is n=5:

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

Number Patterns 1. Increment X

Print the following to size n, following is n =7:

```
0     0
 1   1
  2 2
   3
  4 4
 5   5
6     6
```

Number Patterns 2. Prime Triangle

Print the following to size n, following is n=4:

```
 2
 3  5
 7 11 13
17 19 23 29
```

a. Generate isPrime(n) function from chatgpt and use that (prime, return 1 else return 0)

b. Do it without isPrime(n)

Number Patterns 3. Alternate Increment

Generate for size n, following is n=4:

```
  1  3  5  7
     4  6  8
        7  9
          10
```

Note: u can do it without setw, but use it so that the output is clearer.

Number Patterns 4. Triangular Numbers

Print the following uptil ‘n’ in while:

1, 3, 6, 10, 15, 21, 28, etc.

Number Patterns 5. Dual Square Pattern

Print the following for size N, following is n=5:

1 ***4 3 2 1***

2 3 ***3 2 1***

3 4 5 ***2 1***

4 5 6 7 ***1***

5 6 7 8 9

Number Patterns 6: Empty Pyramid

Print the following for size N, following is n=7:

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

Dry Run 1.

```
for (int i = 0; i < 5; i++) {
   if (i == 2)
    continue;
   cout << i << " ";
 }
```

Dry Run 2.

```
for (int i = 0; i < n; i++) {
   if (arr[i] < 0)
    continue;
   sum += arr[i];
 }
```

Dry Run 3.

```
int i = 0;
 while (i < 5) {
   if (i == 2)
    continue;
   cout << i << " ";
   i++;
 }
```

Dry Run 4.

```
char i = 2;

for(i--; i>0; cout << int(i)){

cout << “*”;

}
```

Dry Run 5.

```
int count=0;

for(;;){

     if (count==10)

          break;

     cout << ++count;

}
```

Dry Run 6.

dry run + test:

```
char i =0;

for (; i++; cout <<i){

     cout << int(i);

}
```

Dry Run 7.

```
#include <iostream>

using namespace std;

int main(){

    int loopvar = 5;

    while (cout << "Hello " && loopvar--){

    }

return 0;

}
```

Alphabet Hourglass

Create an Alphabet Hourglass of size n, following is for n=4:

```
A B C D

 A B C

  A B

   A

  A B

 A B C

A B C D
```

Alphabet X

Create an Alphabet 'X' of size n, following is for n=7:

```
A           A

  B       B
```

```
    C   C

      D

    E   E

  F       F

G           G
```

Concentric Squares

Generate Concentric Squares, outermost square having sides of length n, where n is an odd number:

n=9:

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

*               *

*   * * * * *   *

*   *       *   *

*   *   *   *   *

*   *       *   *

*   * * * * *   *

*               *

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

n=7 :

```
* * * * * * *

*           *

*   * * *   *

*   *   *   *

*   * * *   *

*           *

* * * * * * *
```

Pointers

Helpful Tips:

// The refer and derefer operators (* and &) have the absolute lowest precedence compared to everything else;

Pointer Arithmetic for Array Traversal

Problem:

Write a function that finds the sum of elements in an integer array using pointer arithmetic. Do not use array indexing. You are only allowed to use pointer dereferencing and pointer arithmetic.

```
int sum(int *arr, int n);
```

Implementing strdup (String Duplicate)

Problem:

Write a function my strdup that duplicates a given string by allocating memory and copying the string into the newly allocated memory. The function should return the pointer to the duplicated string.

```
char * strdup(const char *str);
```

- The function should handle NULL input by returning NULL.

Pointer to Pointer Swap

Problem:

Write a function that swaps the values of two integers using a pointer to pointer approach. You are allowed to pass the addresses of two integer variables using pointer to pointer.

```
void swap(int **x, int **y);
```

Test the function by creating two integer variables and printing their values before and after the swap.

Dynamic Memory Allocation for 2D Array

Problem:
Write a program that dynamically allocates memory for a 2D array of integers. The program should also populate the array with values, and then free the allocated memory.

```
int **create2DArray(int rows, int cols);
void fill2DArray(int **arr, int rows, int cols);
void free2DArray(int **arr, int rows);
```

- create2DArray should dynamically allocate memory for the array. - fill2DArray should fill the array with values (for example, set arr[i][j] = i * j). - free2DArray should free the allocated memory.

Function Pointer to Implement Basic Calculator

Problem:
Create a simple calculator program that uses function pointers. The program should perform addition, subtraction, multiplication, and division using function pointers.

```
int add(int a, int b) { return a + b; }
int subtract(int a, int b) { return a - b; }
int multiply(int a, int b) { return a * b; }
int divide(int a, int b) { return a / b; }
```

int (*operation[])(int, int) = {add, subtract, multiply, divide}; Test the program by performing each operation on two integers.

Custom calloc Implementation

Problem:
Write your own implementation of the calloc function. The function should allocate memory for an array of elements and initialize all the bytes to zero.

```
void *my calloc(size t num elements, size t element size);
```

- The function should return a pointer to the allocated memory.
- If either num elements or element size is 0, return NULL.

Find the Maximum Element in an Array using Pointer Arithmetic

Problem:
Write a function that finds the maximum element in an integer array using pointer arithmetic. Do not use array indexing; only pointer dereferencing and pointer arithmetic should be used.

```
int findMax(int *arr, int n);
```

- The function should return the value of the maximum element in the array.

String Concatenation Using Pointers

Problem:
Write a function that concatenates two strings using pointer manipulation. Do not use any built-in string functions like strcat.

```
void my strcat(char *dest, const char *src);
```

- The function should append the src string to the dest string. - Make sure to handle null-terminated strings properly.

Detecting Memory Leaks

Problem:
Write a program that simulates a memory leak by allocating memory but failing to free it. Implement a simple function that detects if memory was allocated but not freed (hint: this will be a simplified detection).

```
void memoryLeak(int size) { // Allocate memory and forget to free it }
int detectLeak() { // Return 1 if leak is detected, otherwise 0 }
```

- The function detectLeak will not actually detect real memory leaks but should simulate checking if memory was allocated without freeing it.

Reversing an Array using Pointer Arithmetic

Problem:
Write a function that reverses an array in place using only pointer arithmetic. Do not use array indices.

```
void reverseArray(int *arr, int n);
```

- The function should reverse the elements of the array arr of size n.

Circular Buffer Implementation Using Pointers

Implement a circular buffer using pointers. The buffer should support the following operations: 1. Adding an element to the buffer (enqueue). 2. Removing an element from the buffer (dequeue). 3. Checking if the buffer is full or empty.
Use the following functions:

```
void initBuffer(CircularBuffer *cb, int size);
void enqueue(CircularBuffer *cb, int value);
int dequeue(CircularBuffer *cb);
int isFull(CircularBuffer *cb);
int isEmpty(CircularBuffer *cb);
```

Use dynamic memory allocation to manage the buffer.

Implementing Linked List Traversal Using Pointers

Problem:
Write a program to implement traversal of a singly linked list using pointers. Each node should store an integer value.

```
typedef struct Node { int data; struct Node *next; } Node;
```

Functions to implement:

```
Node *createNode(int value);
void traverseList(Node *head);
void freeList(Node *head);
```

Ensure proper memory management by freeing all nodes at the end of the program. 1125

Pointer-Based Matrix Multiplication

Problem:
Write a program to multiply two matrices using pointers. Use dynamic memory allocation for the matrices.

```
void matrixMultiply(int **mat1, int rows1, int cols1, int **mat2, int rows2, int cols
```

- Validate that the number of columns in mat1 matches the number of rows in mat2. - Allocate memory for the result matrix dynamically.

Pointer-Based String Reverse

Problem:
Write a function that reverses a string in place using pointer manipulation.

```
                void reverseString(char *str);
```
- The function should handle null-terminated strings and ensure proper memory safety.

Multi-Level Pointer Manipulation

Problem:
  Write a program to demonstrate multi-level pointers by dynamically creating a variable and modifying its value using a triple pointer.

```
            void manipulateTriplePointer(int ***ptr);
```
- Use this program to show the value at the original pointer being changed indirectly.

Dry Run Ult Test

```
#include <iostream>
using namespace std;
int main() {
    int arr[100];
    for(int i = 0; i<100;i++)
        arr[i] = i;
    int *(ptr)=arr;
    int *(ptr2)[3] = {ptr,ptr+1,ptr+2};
    int *ptr3[2][3] = { {ptr, ptr+1, ptr+2},
                        {ptr+3, ptr+4, ptr+5}
                        };
    cout << "6. " << *ptr << endl;
    cout << "7. " << *(ptr + *ptr) << endl;
    cout << "8. " << *(&ptr[4] - 1) << endl;
    cout << "9. " << *(ptr + *(ptr + 2)) << endl;
    cout << "10. " << ptr[ ptr[3] ] << endl;
    cout << "11. " << **ptr2 << endl;
    cout << "12. " << *ptr2[2] << endl;
    cout << "13. " << **(ptr2 + 1) << endl;
    cout << "14. " << *(*(ptr2 + 2)) << endl;
    cout << "15. " << *ptr2[ *ptr ] << endl;
    cout << "16. " << *ptr3[0][0] << endl;
    cout << "17. " << **ptr3 << endl;
    cout << "18. " << *ptr3[1][2] << endl;
    cout << "19. " << **(ptr3[1] + 1) << endl;
    cout << "20. " << *(*(*(ptr3 + 1) + 0)) << endl;
    cout << "21. " << *(*ptr3[0] + 2) << endl;
    cout << "22. " << **(ptr3[0] + *ptr) << endl;
    cout << "23. " << *(*(ptr3 + 1)[0]) << endl;
    cout << "24. " << *(*(*(ptr3) + 1) + 1) << endl;
    cout << "25. " << *(ptr3[ *(ptr) % 2 ][ *(ptr + 1) % 3 ]) <<
endl;

    return 0;
}
```

Dry Run 1.

```
int arr[] = {5, 10, 15};
int* p = arr;
// ++*p+=3;
// *++p += 3;
// comment out either one of these
p++;
cout << arr[0] << " " << arr[1] << " " << arr[2];
```

Dry Run 2.

```
#include <iostream>
using namespace std;

int operate (int a, int b){
    return (a * b);
}
float operate (float a, float b){
    return (a / b);
}
int main (){
    int x = 6, y = 7;
    float s = 4.0, u = 2.0;
    cout << operate (x, y) <<endl<< operate (s, u);
    return 0;
}
```

Dry Run 3.

```
#include <iostream>
using namespace std;
int main() {
    int arr[2] [3]={
            {1,2,3},
            {4,5,6}
            };
    int (*ptr) [3]=arr;
    // cout<<ptr[1] [2];
    return 0;
}
```

Dry Run 4.

```
#include <iostream>
using namespace std;
int* findNull(int* array){
    while(*(array++)!='\0'){}
    return array--;
}
int main() {
    int arr[10]={1, 2, 3, 4, 5, '\0', 1};
    int *nullPtr = findNull(arr);
    while(nullPtr>arr){
        cout << *(--nullPtr) << endl;
    }
    return 0;
}
```

Dry Run 5.

```
char s[8]={'C','+','\n','+','\r','!','\0'};
```

```
cout<<s;
```

Dry Run 6.

```
int arr[] = {2, 4, 6, 8};
int *p = arr;
cout << *p++ << " " << *p++ << " " << *p;
```

Dry Run 7.

```
const int s=3;
int* listMystery(int list[][::s]){
    int i = 1, k=0;
    int *n = new int[::s];
    for(int i=0;i<::s;++i)
        n[i]=0;
    while(i < ::s){
        int j = ::s - 1;
        while(j >= i){
            n[k++]=list[j][i] * list[i][j];
            j = j - 1;
        }
        i = i + 1;
    }
    return n;
}
void displayMystery(int * arr){
    cout<<"[ ";
    for(int i=0;i<::s;++i)
    cout<<arr[i]<<(i!=(::s - 1)?" , ":" ");
    cout<<"]"<<endl;
}
int main(){
    int L[][::s] = {{8, 9, 4}, {2, 3, 4}, {7, 6, 1}};
    int *ptr=listMystery(L);
    displayMystery(ptr);
    delete [] ptr;
    return 0;
}
```

Dry Run 8.

```
#include <iostream>
using namespace std;
int main() {
    int arr[100];
    for(int i = 0; i<100;i++)
        arr[i] = i;
    int *(ptr)=arr;
    cout<< *ptr<<endl;
    int *(ptr2)[3] = {ptr,ptr+1,ptr+2};
    int *ptr3[2][3] = {
                    {ptr, ptr+1, ptr+2},
                    {ptr+3, ptr+4, ptr+5}
                    };
    return 0;
}
```

Dry Run 9.

```
#include <iostream>
using namespace std;
int findNull(int* array){
    while(*(array++)!='\0');
    return array-—;
}
int main() {
    int arr[10] = {1, 2, 3, 4, 5, '\0', 1};
    int *nullptr = findNull(arr);
    while(nullptr>arr)
        cout << *(--nullptr) << endl;
    return 0;
}
```

Dry Run 10.

```
#include <iostream>
using namespace std;
int main()
    const int x = 5;
    int y=10;
    const int ptr=&y;
    cout <<<< ptr<<endl;
    *ptr++;
    cout << ptr;
    ptr=&x;
    cout << ptr;
    return 0;
}
```

Dry Run 11.

```
void allocateMemory(int** ptr) {
    *ptr = new int;
    ++ptr = 42:
}
int main() {
    int* value = nullptr;
    allocateMemory(&value);
    cout <<< &value<<" "<<value<<" "<<*value<< endl;
    delete[] value;
    value= nullptr;
return 0;
}
```

Dry Run 12.

```
#include <iostream>
using namespace std;
int main(){
    char** p= new char*[3];
    *p ="Rashid";
    p[1] = {"Sajid"};
    p[2] = {"Ali"};
    char * tmp; int i, j;
    for( i = 0; i<3; i++)
        for( j=0; *(*(p+i)+j)!='\0'; j++){
            cout << p[3-i-1][j];
            if(*(p+j+1)=='\0') cout <<endl;
        }
    return 0;
}
```

Implicit Casting

```
float x1 = 2.4;
float x2 = 2.3;

if(x1 == 2.4) cout<<"1";
if(x1 == 2.4f) cout<<"2";
if(x2 == 2.3) cout<<"3";
if(x2 == 2.3f) cout<<"4";
```

Increment Operators

1. i++ + i++
2. i++ + ++i
3. ++i + ++i
4. ++i + i++

Global + Static Vars

Dry Run 1.

```
const int s=3;
int* listMystery(int list[][::s]){
int i = 1,k=0;
int *n = new int[::s];
for(int i=0;i<::s;++i)
n[i]=0;
while(i < ::s)
{
int j = ::s - 1;
while(j >= i)
{
n[k++]=list[j][i] * list[i][j];
j = j - 1;
}
i = i + 1;
}
return n;
}
void displayMystery(int * arr){
cout<<"[ ";
for(int i=0;i<::s;++i)
cout<<arr[i]<<(i!=(::s - 1)?" , ":" ");
cout<<"]"<<endl;
}
int main(){
int L[][::s] = {{8, 9, 4}, {2, 3, 4}, {7, 6, 1}};
int *ptr=listMystery(L);
displayMystery(ptr);
delete [] ptr;
return 0;
}
```

Dry Run 2.

```
#include <iostream>
using namespace std;
int x = 5;
int main() {
    int x = 10
    cout <<x<<endl;
    cout << ::x<<endl;
}
```

Bitwise Operators

Check Kth Bit

Take 2 Inputs, a Number n and bit number k, and check if kth bit is 1 or 0.
For e.g. is n=6 and k=2, output is ‘Yes’, but if k=0 output is ‘No’.

Count Ones

Take Decimal Number as Input, and output count of one in the binary of number