Create the n-size dynamic array correctly

#include <iostream>

using namespace std;

int main() {int sum = 0; int n, d; int\* data; cin >> n; data = new int[n];

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

cin >> d; \*(data + i) = d;}

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

sum = sum + \*(data + i);

cout << sum; return 0; }

**Question 4. Write the function Q4() to satisfy all of the following requirements:**

**The function reads from a file called data.txt. In data.txt, each line contains exactly three integers. Each integer can be a value from 0 to 100.Note that the question text does not provide the file data.txt for download.It outputs the total number of distinct values appearing in the file.It outputs the two integers that occur most often in the file. For each of the two integers, also output the number of times it appears in the file.For example, if integer 1 appears 30 times, integer 2 appears 40 times, integer 3 appears 50 times, and integer 4 appears 20 times in data.txt, then the function outputs thefollowing to the console windows:**

Void Q4(){

int frequency[101] = {0}; // keep the frequency of i at frequency[i]

int top[2][2] = {-1}; // row = (i, frequency)

ifstream fin;

int n1, n2, n3;

int total = 0;

fin.open("data.txt");

if (fin.is\_open())

while (!fin.eof()){

fin >> n1 >> n2 >> n3;

if (!fin.fail()){

frequency[n1]++;

frequency[n2]++;

frequency[n3]++;

}

}

fin.close();

for (int i = 0; i <= 100; i++) // # of unique integers in the file

if (frequency[i] > 0) total++;

for (int j = 0; j <= 100; j++){. // top 2 most frequent numbers

if (frequency[j] > top[0][1]){ // larger than top1

top[1][0] = top[0][0]; // move the top1 to top2

top[1][1] = top[0][1];

top[0][0] = j;

top[0][1] = frequency[j];

}

else

if (frequency[j] > top[1][1]){

// larger than top2 but less than top1

top[1][0] = j;

top[1][1] = frequency[j];

}

}

cout << "Total number of integers in the file = "<< total << endl;

cout << "The most frequently occurred number is "<< top[0][0] << " with frequency = " << top[0][1] << endl;

cout << "The second most frequently occurred number is "<< top[1][0] << " with frequency = " << top[1][1] << endl;}

int main(){

Q4();

return 0;}   
**TICTACTOE:**

#include <iostream>

using namespace std;

bool isWin(char game[3][3]) {

bool win = false;

// row

if (game[0][0] == game[0][1] && game[0][1] == game[0][2] && (game[0][0] == 'X' || game[0][0] == 'O')) win = true;

if (game[1][0] == game[1][1] && game[1][1] == game[1][2] && (game[1][0] == 'X' || game[1][0] == 'O')) win = true;

if (game[2][0] == game[2][1] && game[2][1] == game[2][2] && (game[2][0] == 'X' || game[2][0] == 'O')) win = true;

// column

if (game[0][0] == game[1][0] && game[1][0] == game[2][0] && (game[0][0] == 'X' || game[0][0] == 'O')) win = true;

if (game[0][1] == game[1][1] && game[1][1] == game[2][1] && (game[0][1] == 'X' || game[0][1] == 'O')) win = true;

if (game[0][2] == game[1][2] && game[1][2] == game[2][2] && (game[0][2] == 'X' || game[0][2] == 'O')) win = true;

// diagonal

if (game[0][0] == game[1][1] && game[1][1] == game[2][2] && (game[0][0] == 'X' || game[0][0] == 'O')) win = true;

if (game[0][2] == game[1][1] && game[1][1] == game[2][0] && (game[0][2] == 'X' || game[0][2] == 'O')) win = true;

return win;}

int main() {

int i, j;

char game[3][3] = { ' ' }; // Tic-tac-toe

char player1 = 'X';

char player2 = 'O';

bool turn = true; // false for player 1's turn, true for player 2's turn. Player 1 first.

cout << "X = Player 1" << endl << "O = Player 2" << endl;

int n = 0;

while (n < 9) {

turn = !turn; // use the not-operator to change true to false or false to true.

if (turn == false)

cout << "Player 1: ";

else

cout << "Player 2: ";

cout << "Which cell to mark? i:[0..2], j:[0..2]: ";

cin >> i >> j;

if (game[i][j] != ' ') {

cout << "This cell has been marked!" << endl;

turn = !turn; // flip the turn variable so that next round is still the same player

continue;

}

if (turn == false)

game[i][j] = 'X';

else

game[i][j] = 'O';

n = n + 1;

if (isWin(game)) {

cout << "Win!" << endl;

break; // need to terminate the problem

}

}

if (n == 9) // all cells have been inputted above but no winner yet

cout << "Tie!" << endl;

// show the game to console

cout << game[0][0] << " " << game[0][1] << " " << game[0][2] << endl;

cout << game[1][0] << " " << game[1][1] << " " << game[1][2] << endl;

cout << game[2][0] << " " << game[2][1] << " " << game[2][2] << endl;

return 0;}

**2. The following program will output the text shown in the console window below. Rewrite the program so that it never declare any pointer and produce the same computation to change the letters in the array from small letter to capital letter and from capital letter to small letter.**

**Question given :**

int main() {

char course[] = "CS2311 Computer Programming";

char \*ptr = course;

while (\*ptr != '\0') {

if (\*ptr >= 65 && \*ptr <= 90) {

\*ptr = \*ptr++ + 'a' - 'A';

} else if (\*ptr >= 97 && \*ptr <= 122) {

\*ptr = \*ptr++ + 'A' - 'a';

else ptr++;}

ptr = course;

while (\*ptr != '\0') {

cout << \*ptr++;}

return 0;}

**CONSOLE WINDOW : cs2311 COMPUTER pROGRAMMING**

**My Answer:** int main() {

char course[] = "CS2311 Computer Programming";

int i = 0;

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

if (course[i] >= 65 && course[i] <= 90) {

course[i] = course[i] + 'a' - 'A';

} else if (course[i] >= 97 && course[i] <= 122) {

course[i] = course[i] + 'A' - 'a';

i++;

}

i = 0;

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

cout << course[i++];

} return 0;}

**Write a program to accept a positive single-digit odd integer N (e.g., 3, 5, 7, 9) and print a pyramid using numbers with height N.**

**Enter number of rows: 3**

**1**

**2 3 2**

**3 4 5 4 3**

**Enter number of rows: 5**

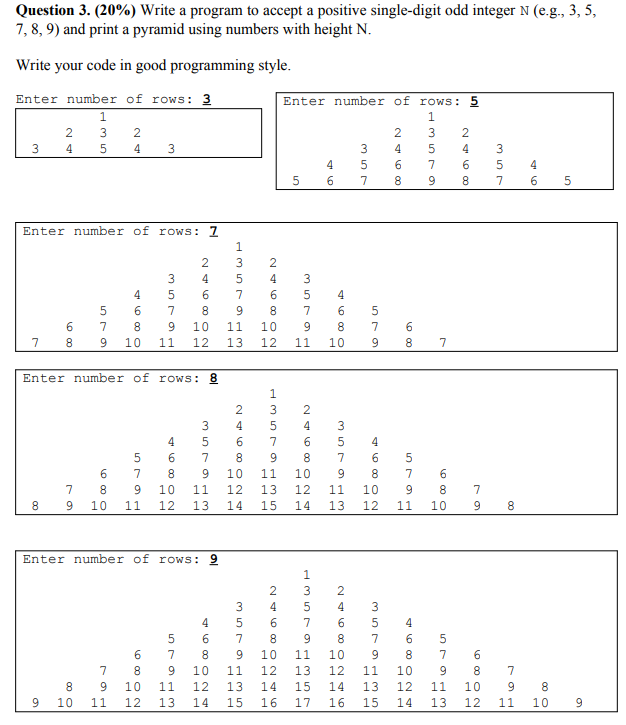
**1**

**2 3 2**

**3 4 5 4 3**

**4 5 6 7 6 5 4**

**5 6 7 8 9 8 7 6 5**



#include <iostream>

using namespace std;

int main() {

int N(0);

cout << "Enter number of rows: ";

cin >> N;

//FIRST LINE

for (int i = 0; i <= N - 1; i++) cout << " ";

cout << 1 << endl;

//BODY 2.. last line

for (int row = 2; row <= N; row++) {

for (int i = 0; i < N - row; i++) cout << " ";

for (int j = row; j <= row \* 2 - 1; j++) {

cout << j ;

if (j >= 9) cout << " ";

else cout<< " ";

}

for (int j = row \* 2 - 1; j >= row; j--) {

cout << j;

if (j >= 10) cout << " ";

else cout << " ";

}

cout << endl;

}

return 0;

}

**Harmonic Mean.**

#include <iostream>

#include <iomanip>

using namespace std;

double H(int n) {

double mean(0);

if (n == 1) return 1;

else {

for (int i = 1; i <= n; i++) {

mean += 1.00 / i;

}

return double(n) / mean;

}

}

int main() {

int N(0), times(0);

double harmonic(0);

do {

cout << "Input a positive odd number (0 to end): ";

cin >> N;

times++;

if (N==0 ) break ;

} while ((N < 0) || (N % 2 == 0)) ; // N is negative or N is even number

If(N>0) {

harmonic = H(N);

cout << fixed << setprecision(4);

cout << "The Harmonic mean is: " << harmonic << endl;

}

cout << "Users have inputted for " << times << " time(s)." << endl;

return 0;

}

QUESTION 5 (20%)

The file luck.txt is a word dictionary, which contains 10,000 lines. Each line contains exactly an English word. Write a program to meet the following requirements:

* Accept one word from console ;
* It determines whether there is any word in the file starting with the input word and output all of them to Console.
* It handles file management properly.
* It keeps the words in the file in an array named source[].
* You are allowed to use <iostream> and <fstream>. **Other libraries are not allowed to use.**
* **It only uses pointers** to update the array source or read from source[], i.e., do not use syntax like source[i] to refer to the element kept in source[i]).

Note: If your program uses such array syntaxes, marks will be deducted.

You may assume that each word contains no more than 20 characters.

#include <iostream>

#include <fstream>

using namespace std;

bool checkword(char w[], char source[][]) { // w[] -- word[]

ifstream fin;

fin.open("luck.txt");

if (!fin.is\_open()) {

cout << "Unable to open file " << endl;

return -10;

}

char line[20];

//declare a dynamic array for source[10000]?

int x(0), index(0);

int len = strlen(w);

while (!fin.eof()) {

fin.getline(line, 20);

index = 0;

for (int y = 0; y < len; y++) {

if (\*(w + y) == \*(line + y));

else break;

if (y == len - 1) {

for (int z = 0; \*(line + z) != '\0'; z++)

\*(source[x] + z) = \*(line + z);

}

x++;

}

}

}

fin.close();

if (x > 0) return true;

else return false;

}

int main() {

char word[20];

cin.getline(word, 20);

char p[10][20];

if (checkword(word, p)) {

for (int i = 0; \*(p[i] + 0) != '\0'; i++) {

for (int i = 0; \*(word + i) != '\0'; i++) {

cout << \*(word + i) ;

}

cout << " is the prefix of ";

//print out the prefix[] "p"

cout << \*(p[i]);

cout << endl;

}

}

else {

for (int i = 0; \*(word + i) != '\0'; i++){

cout << \*(word + i);}

cout << " is not defined in the source file " << endl;

}

return 0;

}

**Write a program that takes two positive integers as input from the user and calculates the greatest common divisor (G.C.D). Please note that the program needs to meet the following requirements:**

* **The program should ask the user to input two positive integers and then output the G.C.D.**
* **Solve the problem with recursion, otherwise, 50% of the scores may be subjective to deduction.**

**HINT: The G.C.D of two positive integers is the greatest common factor that divides them exactly. For example, the G.C.D of 15 and 10 is 5.**

**GCD**

#include<iostream>

using namespace std;

int hcf(int n1, int n2);

int main() {

int n1, n2;

cout << "Enter two positive integers: " << endl;

cin >> n1 >> n2;

cout << "G.C.D is " << hcf(n1,n2);

return 0;

}

int hcf(int n1, int n2) {

if (n2 != 0)

return hcf(n2, n1 % n2);

else

return n1;

}**Given an integer n, return the number of ways you can write n as the sum of consecutive positive integers. Only one integer (n itself) is also a way. Note that 1 <= n <= 109 and you must write a function consecutiveNumbersSum to solve this problem. The function prototype is provided.**

#include <iostream>

using namespace std;

int consecutiveNumbersSum(int N);

int main() {

int n, res;

cin >> n;

res=consecutiveNumbersSum(n);

cout << res << endl;

return 0;

}

int consecutiveNumbersSum (int N) {

int a, k, count = 0;

for (a = 1; a <= N; a++) {

for (k = 1; k <= N; k++) {

int tmp = k \* (2 \* a + k - 1);

if (tmp == 2 \* N)

count++;

else if (tmp > 2 \* N)

break;

}

}

return count;

}

**Q1 (change to j)**

#include<iostream>

using namespace std;

int main() {

int r, x(2);

char y = 5;

**char c = 'A' - 'Z' + 'A'+65 ;**

r = y % x;

c += r;

cout << c;

**Q2(array size too limited if input 3 )**

#include<iostream>

using namespace std;

void main() {

**char s[100] = "C"**; cin >> s;

int i; cin >> i;

cout << s[i];

i++;

cout << s << ".";

}

**Q3(change to 32 )**

#include<iostream>

using namespace std;

int main() {

int a[] = { 1, 2 }, \* p = a;

int b = \*(p + 1); // b = 2

**int c = 1;**  // Set c = 1 to get desired output

int d = b + c; // d = 3

cout << d << b; // Output: 32

return 0;

}

**Q4(4 x Hello 6 x Hello 8 x Hello)**

#include<iostream>

using namespace std;

int main() {

int x = 3;

while (x < 10) {

**if (x % 2 == 0)**

cout << x << " x Hello" << endl;

x++;

}

return 0;

}

**Q5(4 7)**

#include <iostream>

using namespace std;

void SwapandAdd(int m, int n) {

int x = m + 1;

m = n + 1;

n = x + 1;

}

int main() {

int x = 4; int y =7;

SwapandAdd(x, y);

cout << x << " " << y;

return 4;

}

**Q6(omomo/ omom)**

#include <iostream>

using namespace std;

char maze[] = "CS2311IsComputerProgramming";

int k = 0;

void search(int i, char c) {

if (maze[i] > c && k <= 4) {

cout << maze[i]; k++;

**search(i - 1, c);**

search(i + 1, c);

}

}

int main() {

search(9, 'Z');

return 0; }

**Q7(29)**

#include<iostream>

using namespace std;

class dummy {

public:

int x;

dummy(int i) {

x = i \* i;

}

void add(int i) {

x = x + i;

}

int isLarger(dummy o, int c) {

if **(x <= o.x + c)** { // Fixed: Compare with o.x instead of x

add(c);

return x;

}

else return x;

}

};

int main() {

dummy a(5), b(-5);

int c = 2;

a.add(c);

cout << "Result: " << a.isLarger(b, c) << endl; // Added a call to isLarger and output

return 0;

}**Q8(Hello)**

#include<iostream>

#include <fstream>

using namespace std;

int main() {

ofstream o;

ifstream i;

char s[60];

o.open("hello.txt");

o << "hello";

o.is\_open();

**o.close();**

i.open("hello.txt");

i >> s;

cout << s;

i.close();

**Q9(2)**

#include<iostream>

using namespace std;

int main() {

char c = 'S';

**int i = 1;** // Start at 1 instead of 0

char exam[] = "CS2311 2021/2022 Exam";

char\* a = exam;

while (i >= 0) {

if (a[i - 1] == c) // Now checks a[0], a[1], etc.

break;

**i++;**

}

cout << i; // Outputs 2 (since a[1] = 'S')

return 0;

}

**Q10(3 NoYes output No)**

#include<iostream>

using namespace std;

int main() {

int k = 0, i = 0;

cin >> k;

int r = k;

char\* p = new char[k];

char q[50];

cin >> q; //q="NoYes"

while (k > 1 && q[i] != 0) {

p[i] = q[i];

i++; k--;

}

**if (i > k) p[i] = '\0';**

cout << p;

}

**CheckPrimeProgramme**

#include <iostream>  
using namespace std;  
 bool isPrime(int num) {  
if (num <= 1) return false;  
for (int i = 2; i \* i <= num; i++) {  
if (num % i == 0) return false;  
}  
return true;  
}  
int main() {  
char choice;  
do {  
cout << "Do you want to check if a number is prime? Enter 'Y' if yes, otherwise the program will quit: ";  
cin >> choice;

if (choice == 'Y') {  
int number;  
cout << "Enter a positive number: ";  
cin >> number;  
if (isPrime(number)) {  
cout << "This is a prime number." << endl;  
} else {  
cout << "This is not a prime number." << endl;  
}  
}  
} while (choice == 'Y');  
cout << "The program quits now." << endl;  
return 0;

**Finduniqleelement or array**

// Function to find the unique element  
int findUniqueElement(int a[], int n) {  
// Sort the array  
for (int i = 0; i < n - 1; i++) {  
int minIndex = i;  
for (int j = i + 1; j < n; j++) {  
if (a[j] < a[minIndex]) {  
minIndex = j;  
}  
}  
// swap two variables  
int temp;  
temp = a[minIndex];  
a[minIndex] = a[i];  
a[i] = temp; //( swap(a[i], a[minIndex]); also correct )  
}  
// Traverse through the sorted array  
for (int i = 0; i < n - 1; i += 2) {  
// If current element is not equal to the next one, it's the unique element  
if (a[i] != a[i + 1]) {  
return a[i];  
}  
}  
return a[n - 1];// If we reach the last element, it must be the unique one  
}  
**Write a program to check if a 9x9 array is a valid Sudoku.**

#include <iostream>

using namespace std;

// print number as word: [4 marks]

void printNumberAsWord(int num) {

switch(num) {

case 1: cout << "first"; break;

case 2: cout << "second"; break;

case 3: cout << "third"; break;

case 4: cout << "fourth"; break;

case 5: cout << "fifth"; break;

case 6: cout << "sixth"; break;

case 7: cout << "seventh"; break;

case 8: cout << "eighth"; break;

case 9: cout << "nineth"; break;

}

}

int main() {

int row, col;

// convert user input to 9x9 array: [6 marks]

char digit[9][10];

cout << "Enter nine 9-digit numbers:\n";

for (row = 0; row < 9; row++) {

cin.getline(digit[row], 10);

for (col = 0; col < 9; col++) {

digit[row][col] -= '0';

}

}

bool valid = true;

// check rows: [4 marks]

for (row = 0; row < 9; row++) {

bool flag[10] = {0};

for (col = 0; col < 9; col++) {

if (!flag[digit[row][col]])

flag[digit[row][col]] = true;

else {

cout << "The ";

printNumberAsWord(row+1);

cout << " row is invalid\n";

valid = false;

break;

}

}

}

// check columns: [4 marks]

for (col = 0; col < 9; col++) {

bool flag[10] = {0};

for (row = 0; row < 9; row++) {

if (!flag[digit[row][col]])

flag[digit[row][col]] = true;

else {

cout << "The ";

printNumberAsWord(col+1);

cout << " column is invalid\n";

valid = false;

break;

}

}

}

// check sub-box: [8 marks]

for (int subbox = 0; subbox < 9; subbox++) {

bool flag[10] = {0};

for (int n = 0; n < 9; n++) {

row = 3\*(subbox/3) + (n/3);

col = 3\*(subbox%3) + (n%3);

if (!flag[digit[row][col]])

flag[digit[row][col]] = true;

else {

cout << "The ";

printNumberAsWord(subbox+1);

cout << " sub-box is invalid\n";

valid = false;

break;

}

}

}

// output when valid: [2 marks]

if (valid)

cout << "Valid\n";

return 0;

}

**Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M**

#include <iostream>  
using namespace std;  
void intToRoman (int input) {  
// conversion table definition [8 marks]  
char \*thousands[] = {"", "M", "MM", "MMM", "MMMM"};  
char \*hundreds[] = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"};  
char \*tens[] = {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"};  
char \*units[] = {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"};  
// conversion [6 marks]  
int th = input / 1000;  
input %= 1000;  
int h = input / 100;  
input %= 100;  
int t = input / 10;  
int u = input % 10;  
cout << "\"" << thousands[th] << hundreds[h] << tens[t] << units[u] << "\"" << endl;  
}  
int main() {  
int num;  
// take user input [2 marks]  
cout << "Please input the number for conversion: ";  
cin >> num;  
intToRoman(num);  
return 0;  
}

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**bool isPalindrome(char\* s, int n); //check if string s contains n characters is palindrome**

**bool hasPalindrome(char\* s); //check if string s contains ANY palindrome substrings**

**int main() {**

**char s[256];**

**cout << "Enter a string no longer than 255 characters:\n";**

**cin >> s;**

**if (hasPalindrome(s))**

**cout << "It has palindrome substring(s).";**

**else**

**cout << "It doesn't have any palindrome substring.";**

**return 0;**

**}**

**bool isPalindrome(char\* s, int n) {**

**char \*p, \*q;**

**for (p = s, q = s + n - 1; [2 marks] p < q; p++, q-- [2 marks]) {**

**if (\*p != \*q) [2 marks]**

**return false;**

**}**

**return true;**

**}**

**bool hasPalindrome(char\* s) {**

**int len = strlen(s);**

**int i, j;**

**for (i = 0; i < len; i++) {**

**for (j = i + 1; j < len; j++) {**

**if (isPalindrome(s + i [2 marks], j - i + 1 [2 marks]))**

**return true;**

**}**

**}**

**return false;**

**}**

**We have a program that reads the coordinates of two  
points and calculates the perimeter and area of the  
rectangle formed by the coordinates of these two  
points. We specify that the point coordinates of the  
upper left corner and the lower right corner of the  
rectangle are 1 1( , )A x y and 2 2( , )B x y , respectively.**

#include <iostream>  
#include <iomanip>  
#include <cmath>  
using namespace std;  
class Point {  
private: [1 mark]x  
double x, y;  
public: [1 mark]x  
Point(double \_x, double \_y) : x(\_x), y(\_y) [1 mark]x {};  
~Point();  
double disX(const Point &b) { return b.x – x; [1 mark]x }  
double disY(const Point &b) { return b.y – y; [1 mark]x }  
};

double Rectangle {  
private: [1 mark]x  
Point a, b;  
public: [1 mark]x  
Rectangle(Point \_a, Point \_b) : a(\_a), b(\_b) [1 mark]x {};  
~Rectangle();  
double getS() {  
// double abs(double x); provided by <cmath> calculates the absolute value of input x  
return abs( a.disX(b) \* b.disY(b) [1 mark]x );  
}  
double getP() {  
return abs( 2\*(a.disX(b)+abs(a.disY(b)) [1 mark]x );  
}  
}  
int main() {  
cout << "Input (x, y) coordinates for Point A and B: ";  
double ax, ay, bx, by;  
cin >> ax >> ay >> bx >> by;  
if ( ax > bx || ay < by || cin.fail() [2 mark]x ) {  
cout << "The input points are invalid.\n";  
}  
else {  
Point a(ax, ay) [1 mark]x ;  
Point b(bx, by) [1 mark]x ;  
Rectangle rect(a, b) [1 mark]x ;  
cout << fixed << setprecision(2) [1 mark]x ;  
cout << "The area of the rectangle is ";  
cout << rect.getS() [1 mark]x << endl;  
cout << "The perimeter of the rectangle is ";  
cout << rect.getP() [1 mark]x << endl;  
return 0;  
}  
}

**Q1(000000000000)**

#include <iostream>

using namespace std;

int main() {

int X[3][4] = { 0 };

for (int i = 0; i < **3**; i++)

for (int j = 0; j < 4; j++)

cout << X[i][j];

return 0;

}

**Q2(in 22 out 2 )**

#include <iostream>

using namespace std;

int main() {

char c[2] = { 1, 1 };

while (c[1] != '\0') {

cin.getline(c, **2**);

cout << c << endl;

}

return 0;

}

**Q3(CS2311 )**

#include using namespace std; int main() { char s**[20**] = "Hello World"; cin >> s; int i = 0; while (s[i] != 0) { cout << s[i]; i++; } cout << endl << s; return 0; }

**Q4(out 18 )**

#include #include using namespace std; int main() { char l[60]; ifstream f; int i(0); f.open("Q1.cpp"); if (f.is\_open()) { f >> l; while **(!f.eof()**) { f.getline(l, 60); i++; } } cout << i; f.close(); return 0; }

**Q5(OK Erorr(1) Erorr(0)**

#include using namespace std; class X { int a; public: X() { a = 0; } X(int i) { a = i; }; void print() { cout << "OK" << endl; } void print2() {cout << "Erorr(" << a << ")" << endl;} void isEqual(X another) { if **(a == another.a**) print(); else print2(); } }; int main() { X x[2]; X x0(0), x1(1); x0.isEqual(x[0]);

**Q6(57 )**

#include using namespace std; #define SIZE 9 int main() { int s[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; int\* p = &s[SIZE / **2**]; cout << \*p; p = p + s[SIZE / p[0]]; cout << \*p; return 0; }

**Q7(Found at 8**

#include using namespace std; #define bound 10 char sorted[] = { "0123456789" }; bool visited[bound] = { false }; int Up(int x) { return (x + 1) % bound; } int Down(int x) { return (x - 1) % bound; } void search(char c, int x) { if (visited[x]) return; else visited[x] = true; if (sorted[x] == c) { cout << "Found at " << x << endl; } else if (**sorted[x] < c**) search(c, Up(x)); else search(c, Down(x)); } int main() {

**If the purchase is between $1,000 and $2,999 inclusively, there is a 10% discount.  
2. If the purchase is between $3,000 and $4,999 inclusively, there is a 10% discount on  
the first $2,999 purchase, and the rest of the purchase amount will get a 20%  
discount.  
3. If the purchase is between $5,000 and $9,999 inclusively, there is a 10% discount on  
the first $2,999 purchase, a 20% discount for the next $2,000, and a 30% discount for  
the rest of the purchase amount.  
4. Finally, if the total purchase is $10,000 or over, the whole purchase will get a 25%  
discount.**

if (purchase <= 0){  
cout << "Invalid input!" << endl;}  
else {  
discount = 0; // for range [0, 999]  
if (purchase >= 10000) // for range [10000, +inf]  
discount = purchase \* 0.25;  
else if (purchase >= 5000) // for range [5000, 9999]  
discount = (purchase - 4999)\* 0.3 + 2000 \* 0.2 + 2999 \* 0.1;  
else if (purchase >= 3000) // for range [3000, 4999]  
discount = (purchase - 2999)\*0.2 + 2999 \* 0.1;  
else if (purchase >= 1000) // for range [1000, 2999]  
discount = purchase \* 0.1;

**The following program reads a positive integer from the Console  
input and displays the revered integer with all zero(es) removed.  
• E.g., if the input is 6789, the program outputs 9876.  
• E.g., if the input is 1230, the program outputs 321.  
• E.g., If the input is 1023, the program outputs 321.  
Complete the following program using C++ iterations. You cannot  
use any additional libraries or functions.**

#include <iostream>

using namespace std;

int main()

{

int n, ans=0;

cout << "Input the integer: ";

cin>> n;

while (n){

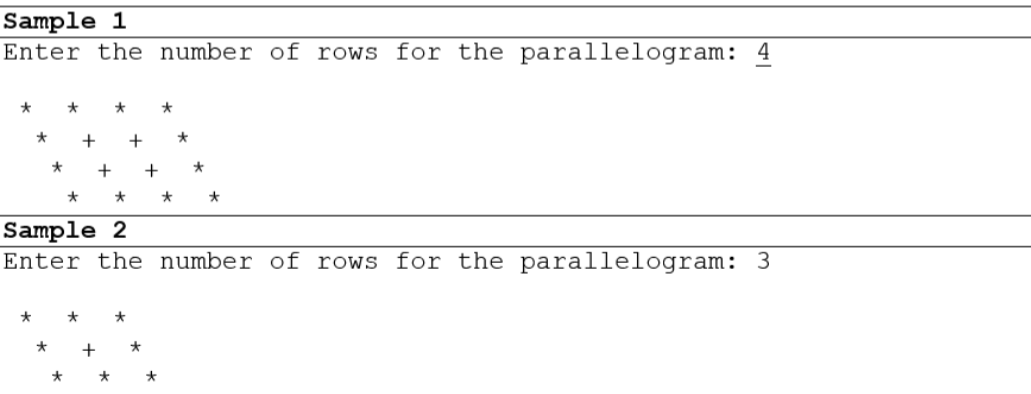
if (n%10)

ans = ans\*10 + n%10; n/=10;

}

cout << "The answer is "<< ans;

return 0;

}

#include <iostream>  
using namespace std;  
int main() {  
int N = 0; // number of rows of the parallelogram  
cout << "Enter the number of rows for the parallelogram: ";  
cin >> N;  
cout << endl;  
for (int i = 0; i < N; i++) {  
 for (int j = 0; j < i; j++)  
 cout << " ";  
 for (int j = 0; j < N; j++)  
 if (i == 0 || i == (N - 1) || j == 0 || j == (N - 1))  
 cout << " \* ";  
 else  
 cout << " + ";  
cout << endl;  
}  
return 0;  
}

**Write a program to accept an integer as a threshold T and a list of**

**integers L from the Console input. The list will end if the value of 0**

**is input. The program should output the sum of the integers in L for**

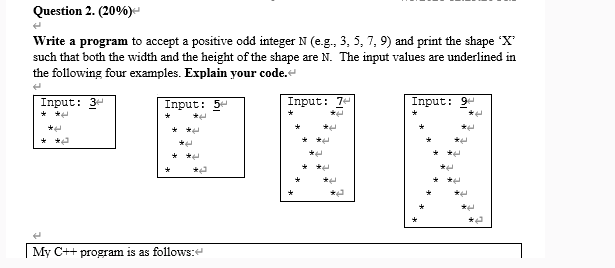
**all these integers that are strictly larger than T. For instance, if T =**

**5 and L is 1 2 3 4 5 6 7 8 9 10 0, then only the integers 6 7 8 9 10 in**

**the list are larger than 5 and their sum is 40. Therefore, the program**

**should output 40.**

#include <iostream>  
using namespace std;  
int main() {  
int threshold;  
cout << "Input the threshold: ";  
cin >> threshold;  
cout << "Input a sequence of integers (input 0 to end): ";  
int n=0;  
int sum = 0;  
do {  
cin >> n;  
if (n > threshold)  
sum = sum + n;  
} while (n != 0);  
cout << "The sum of integers whose are larger than the threshold is "  
<< sum << endl;  
return 0;  
}



#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter a positive odd integer: ";

cin >> N;

if (N % 2 == 0 || N <= 0) {

cout << "Invalid input. N must be a positive odd integer." << endl;

return 1;

}

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

for (int j = 0; j < N; ++j) {

if (j == i || j == N - 1 - i)

cout << "\*";

else

cout << " ";

}

cout << endl;

}

return 0;

}

The file **dictionary.txt** is a word dictionary, which contains 1000 lines, and each line contains exactly one English word. Write a program to meet the following requirements:

* It determines whether the input word is defined in the word list kept in the dictionary.
* It handles file management properly.
* It keeps the words in the dictionary in an array named dict[].
* It does not use any string library function (e.g., strcmp, strncmp).
* It only uses pointers to update dict or read from dict[] (i.e., do not use syntax like dict[i] to refer to the element kept in dict[i]).

Note: If your program uses such array syntaxes, marks will be deducted.

You may assume that each word contains no more than 20 characters.

dictionary.txt

hello

world

cs2311

computer

programming

Example 1

hello

hello is in the dictionary.

Example 2

cs2311

cs2311 is in the dictionary.

Example 3

Hello

Hello is not in the dictionary.

#include <iostream>

#include <fstream>

using namespace std;

int main() {

const int MAX\_WORDS = 1000;

const int MAX\_LEN = 21; // 20 chars + null terminator

char dict[MAX\_WORDS][MAX\_LEN];

char\* ptrDict[MAX\_WORDS];

char input[MAX\_LEN];

int count = 0;

ifstream fin("dictionary.txt");

if (!fin) {

cout << "Failed to open dictionary.txt" << endl;

return 1;

}

while (fin.getline(dict[count], MAX\_LEN)) {

ptrDict[count] = dict[count];

count++;

if (count >= MAX\_WORDS) break;

}

fin.close();

cout << "Enter a word: ";

cin.getline(input, MAX\_LEN);

bool found = false;

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

char\* p = input;

char\* q = ptrDict[i];

bool match = true;

while (\*p || \*q) {

if (\*p != \*q) {

match = false;

break;

}

p++; q++;

}

if (match) {

found = true;

break;

}

}

if (found)

cout << input << " is in the dictionary." << endl;

else

cout << input << " is not in the dictionary." << endl;

return 0;

}

**\*\*EXAMPLE-1:\*\***

**PLEASE INPUT A POSITIVE INTEGER N: 4**

**PLEASE INPUT A POSITIVE INTEGER M NOT GREATER THAN N: 3**

**The permutation is 24**

**The combination is 4**

**The ratio is 6.00**

**\*\*EXAMPLE-2:\*\***

**PLEASE INPUT A POSITIVE INTEGER N: 5**

**PLEASE INPUT A POSITIVE INTEGER M NOT GREATER THAN N: 4**

**The permutation is 120**

**The combination is 5**

**The ratio is 24.00**

**\*\*EXAMPLE-3:\*\***

**PLEASE INPUT A POSITIVE INTEGER N: -1**

**PLEASE INPUT A POSITIVE INTEGER N: 2**

**PLEASE INPUT A POSITIVE INTEGER M NOT GREATER THAN N: 3**

**PLEASE INPUT A POSITIVE INTEGER M NOT GREATER THAN N: 1**

**The permutation is 2**

**The combination is 2**

**The ratio is 1.00**

#include <iosteam>

double factorial(int n) {

    double fact = 1.0;

    for(int i = 1; i <= n; i++) {

        fact \*= i;

    }

    return fact;

}

double permutation(int n, int m) {

    return factorial(n) / factorial(n - m);

}

double combination(int n, int m) {

    return factorial(n) / (factorial(m) \* factorial(n - m));

}

int main() {

    int n, m;

    do {

        cout << "Please input a positive integer n: ";

        cin >> n;

    } while (n <= 0);

    // Input for m with validation

    do {

        cout << "Please input a positive integer m: ";

        cin >> m;

        if (m > n) {

            cout << "Please input a positive integer m not larger than n: ";

        }

    } while (m <= 0 || m > n);

    double perm = permutation(n, m);

    double comb = combination(n, m);

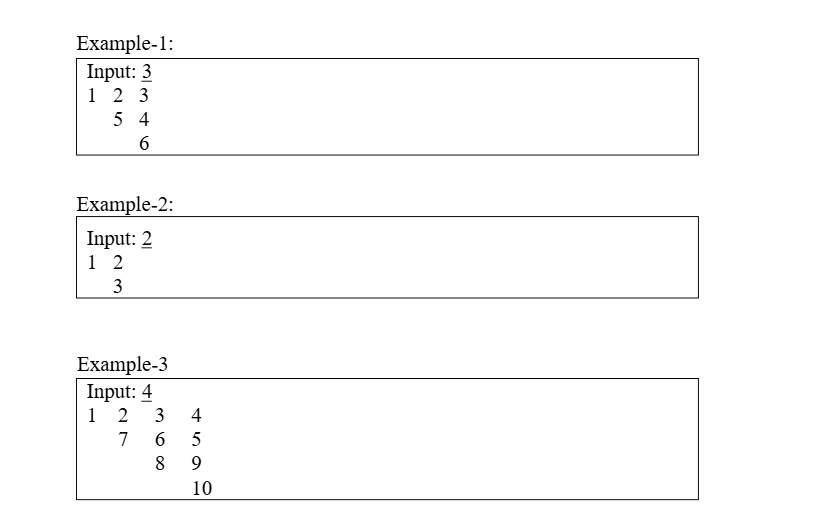
    double ratio = perm / comb;

    cout << "The permutation is " << perm << endl;

    cout << "The combination is " << comb << endl;

    cout << "The ratio is " << fixed << setprecision(2) << ratio << endl;

    return 0;

#include <iostream>  
using namespace std;  
int main() {  
 int n;  
 cout << "Input: ";  
 cin >> n;  
 int count = 1;  
 for (int i = 0; i < n; i++) {  
 for (int t = 0; t < i; t++) {  
 cout << "\t";  
 }  
 int start = count;  
 int end = count + (n - i) - 1;  
 if (i % 2 == 0) {  
 for (int j = start; j <= end; j++) {  
 if (j != start) cout << "\t";  
 cout << j;  
 }  
 } else {  
 for (int j = end; j >= start; j--) {  
 if (j != end) cout << "\t";  
 cout << j;  
 }  
 }  
 cout << endl;  
 count = end + 1;  
 }  
 return 0;  
}

With recursion, derive whether the digits in integer P is a sub-sequence of the digits in another integer N. i.e. derive whether N contains all the digits in P in order (but not necessarily continuous) . You may assume that both numbers are positive. [12 marks] In this question, students are not allowed to use loop, array or any library. Input Output Explanation 12345 24 Y 12345 contains 2 & 4 and 2 comes before 4. 12345 42 N Although 12345 contains 4 & 2, but 2 comes first. 12345 22 N 12345 contains only a single 2. 12345 12345 Y 12345 contains all digits in correct order. 1 1234567890 N N contains only one digit.

#include <iostream>  
using namespace std;  
bool check(int N, int P) {  
 if (P == 0) return true;  
 if (N == 0) return false;  
 if (N % 10 == P % 10) {  
 return check(N / 10, P / 10);  
 } else {  
 return check(N / 10, P);  
 }  
}  
int main() {  
 int N, P;  
 cin >> N >> P;  
 if (check(N, P)) {  
 cout << "Y" << endl;  
 } else {  
 cout << "N" << endl;  
 }  
 return 0;  
}

. Implement a function called Cap() which format the input cstring as specified [20 marks]:  The function allocates memory for the output cstring and the start pointer is returned.  The function is not allowed to change the content of the input cstring.  The function treats all non-alphabet character as space and alphabet sequence as word.  The output cstring contains no leading and trailing spaces and the word(s) are separated by exactly one space character.  The words will be rendered in lowercase, except for the first and last character of each word, which is in uppercase.  Students are not allowed to use any library in the solution

#include <iostream>  
using namespace std;  
  
char \*Cap(char \*S) {  
 int len = 0;  
 while (S[len] != '\0') len++;  
 char \*out = new char[len \* 2 + 1]; // Allocate more than enough space  
 int i = 0, j = 0;  
 bool inWord = false;  
 int wordStart = 0;  
 while (i <= len) {  
 char c = S[i];  
 bool isAlpha = (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');  
 if (isAlpha) {  
 if (!inWord) {  
 if (j != 0) {  
 out[j++] = ' ';  
 }  
 wordStart = j;  
 inWord = true;  
 }  
 if (c >= 'A' && c <= 'Z') {  
 c = c + ('a' - 'A'); // to lowercase  
 }  
 out[j++] = c;  
 } else {  
 if (inWord) {  
 if (j - wordStart >= 1) {  
 if (out[wordStart] >= 'a' && out[wordStart] <= 'z')  
 out[wordStart] = out[wordStart] - ('a' - 'A');  
 if (out[j-1] >= 'a' && out[j-1] <= 'z')  
 out[j-1] = out[j-1] - ('a' - 'A');  
 }  
 inWord = false;  
 }  
 }  
 i++;  
 }  
 if (j > 0 && out[j-1] == ' ') {  
 j--;  
 }  
 out[j] = '\0';  
 return out;  
}  
int main() {  
 cout << "[" << Cap((char\*)"hELLo123") << "]\n"; // [HellO]  
 cout << "[" << Cap((char\*)"?? thIS is~ a 'TEST'??!") << "]\n"; // [ThiS IS A TesT]  
 return 0;  
}

**Q2Enter integers (-1 to stop): 1 2 3 4 5 -1**

**Array after doubling:**

**2 4 6 8 10**

#include <iostream>

using namespace std;

void readArray(int arr[], int& size) {

size = 0;

int temp;

while (cin >> temp && temp != -1 && size < 50) {

arr[size++] = temp;

}

}

void doubleArray(int arr[], int size) {

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

arr[i] \*= 2;

}

}

void printArray(const int arr[], int size) {

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

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

if ((i + 1) % 10 == 0)

cout << endl;

}

cout << endl;

}

int main() {

int arr[50];

int size;

cout << "Enter integers (-1 to stop): ";

readArray(arr, size);

doubleArray(arr, size);

cout << "Array after doubling:\n";

printArray(arr, size);

return 0;

}

**Q3Develop a program that keeps asking the user to enter a number until:**

* **The user enters a number divisible by both 3 and 5.**

**Enter a number: 4**

**Enter a number: 7**

**Enter a number: 15**

**Total numbers entered: 3**

**Sum of numbers: 26**

#include <iostream>

using namespace std;

int main() {

int num, count = 0, sum = 0;

do {

cout << "Enter a number: ";

cin >> num;

sum += num;

count++;

} while (!(num % 3 == 0 && num % 5 == 0));

cout << "Total numbers entered: " << count << endl;

cout << "Sum of numbers: " << sum << endl;

return 0;

}

Q4**(Shape Printing)**

**Write a program to print the following pattern based on input n:**

**Example for n = 4:**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**Then in reverse:**

**1 2 3**

**1 2**

**1**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter n: ";

cin >> n;

for (int i = 1; i <= n; ++i) {

for (int j = 1; j <= i; ++j) {

cout << j;

if (j != i) cout << " ";

}

cout << endl;

}

for (int i = n - 1; i >= 1; --i) {

for (int j = 1; j <= i; ++j) {

cout << j;

if (j != i) cout << " ";

}

cout << endl;

}

return 0;

}

**Q5Enter number of elements: 5**

**Enter elements: 1 9 3 7 5**

**Maximum = 9**

**Minimum = 1**

**Average = 5.00**

#include <iostream>

using namespace std;

void findStats(const int\* arr, int size, int& maxVal, int& minVal, double& avg) {

maxVal = arr[0];

minVal = arr[0];

int sum = 0;

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

if (arr[i] > maxVal) maxVal = arr[i];

if (arr[i] < minVal) minVal = arr[i];

sum += arr[i];

}

avg = static\_cast<double>(sum) / size;

}

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

int\* arr = new int[n];

cout << "Enter elements: ";

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

cin >> arr[i];

}

int maxVal, minVal;

double avg;

findStats(arr, n, maxVal, minVal, avg);

cout << "Maximum = " << maxVal << endl;

cout << "Minimum = " << minVal << endl;

cout << "Average = " << avg << endl;

delete[] arr;

return 0;

}

**Q6 25**

**30**

**45**

**12**

**60**

**75**

**Write a program that:**

**Reads all integers from the file into a static array (up to 100 numbers).**

**Counts how many numbers are divisible by 5.**

**If more than half are divisible by 5, output "PASS", otherwise "FAIL".**

**If the file cannot be opened, print "Error opening file".**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream fin("data.txt");

if (!fin) {

cout << "Error opening file" << endl;

return 1;

}

int arr[100];

int\* p = arr;

int count = 0;

int divisibleBy5 = 0;

int num;

while (fin >> num && count < 100) {

\*(p + count) = num;

if (num % 5 == 0)

divisibleBy5++;

count++;

}

if (divisibleBy5 > count / 2)

cout << "PASS" << endl;

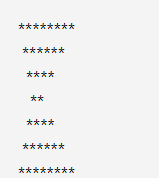
else

cout << "FAIL" << endl;

fin.close();

return 0;

}



#include <iostream>

using namespace std;

void printRow(int spaces, int stars) {

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

cout << " ";

}

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

cout << "\*";

}

cout << endl;

}

void drawPatternHelper(int n, int currentRow) {

if (currentRow == n) return;

printRow(currentRow, 2 \* (n - currentRow));

drawPatternHelper(n, currentRow + 1);

printRow(currentRow, 2 \* (n - currentRow));

}

void drawPattern(int n) {

drawPatternHelper(n, 0);

}

int main() {

int n = 4;

drawPattern(n);

return 0;

}

**Enter a number: 3**

**Odd**

**Sum: 3**

**Enter a number: 4**

**Even**

**Sum: 7**

**Enter a number: -5**

**Warning: Negative number entered!**

**Sum: 2**

**Enter a number: -1**

**Terminating program.**

#include <iostream>

using namespace std;

int main() {

int num, sum = 0;

while (true) {

cout << "Enter a number: ";

cin >> num;

if (num == -1) {

cout << "Terminating program." << endl;

break;

}

if (num < 0) {

cout << "Warning: Negative number entered!" << endl;

}

if (num % 2 == 0) {

cout << "Even" << endl;

} else {

cout << "Odd" << endl;

}

sum += num;

cout << "Sum: " << sum << endl;

}

return 0;

}**Alice**

**Bob**

**Charlie**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream fin("students.txt");

string name;

bool found = false;

while (getline(fin, name)) {

if (name == "Bob") {

found = true;

break;

}

}

if (found)

cout << "Bob found" << endl;

else

cout << "Bob not found" << endl;

return 0;

}

**Take user input for array elements**

**Sort ascending ✅**

**Sort descending ✅**

**Reverse the array ✅**

**Print all results clearly ✅**

#include <iostream>

using namespace std;

void inputArray(int arr[], int size) {

cout << "Enter " << size << " elements: ";

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

cin >> arr[i];

}

void printArray(int arr[], int size) {

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

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

cout << endl;

}

void sortAscending(int arr[], int size) {

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

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

if (arr[i] > arr[j]) {

swap(arr[i], arr[j]);

}

}

}

}

void sortDescending(int arr[], int size) {

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

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

if (arr[i] < arr[j]) {

swap(arr[i], arr[j]);

}

}

}

void reverseArray(int arr[], int size) {

for (int i = 0; i < size/2; i++) {

swap(arr[i], arr[size-1-i]);

}

}int main() {

int size;

cout << "Enter number of elements: ";

cin >> size;

int\* arr = new int[size];

inputArray(arr, size);

cout << "\nOriginal array: ";

printArray(arr, size);

sortAscending(arr, size);

cout << "\nSorted ascending: ";

printArray(arr, size);

sortDescending(arr, size);

cout << "\nSorted descending: ";

printArray(arr, size);

reverseArray(arr, size);

cout << "\nReversed order: ";

printArray(arr, size);

delete[] arr; // Clean up

return 0;

**}Write a program that: Repeatedly asks the user to enter positive numbers Stops when the user enters 0 Calculates and displays: The sum of all numbers The average of all numbers The largest number entered**

#include <iostream>using namespace std;

int main() {

int num, sum = 0, count = 0, max = 0;

while (true) {

cout << "Enter a positive number (0 to stop): ";

cin >> num;

if (num == 0) break;

if (num < 0) continue;

sum += num;

count++;

if (num > max) max = num;

}

cout << "Sum: " << sum << endl;

cout << "Average: " << (count ? (double)sum/count : 0) << endl;

cout << "Largest: " << max << endl;

return 0;

}

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

int main() {

int height;

cout << "Enter height: ";

cin >> height;

for (int i = 1; i <= height; i++) {

for (int j = 0; j < height-i; j++) cout << " ";

for (int j = 0; j < 2\*i-1; j++) cout << "\*";

cout << endl;

} return 0;}

**Let user decide the size of array , and print the array and double the size of array**int main() {  
 int size;  
 cout << "Enter array size: ";  
 cin >> size;  
 int\* arr = new int[size];  
 cout << "Enter " << size << " numbers: ";  
 for (int i = 0; i < size; i++) cin >> arr[i];  
 int newSize = size \* 2;  
 int\* newArr = new int[newSize];  
 for (int i = 0; i < size; i++) newArr[i] = arr[i];  
 for (int i = size; i < newSize; i++) newArr[i] = 0;  
 cout << "Resized array: ";  
 for (int i = 0; i < newSize; i++) cout << newArr[i] << " ";  
 delete[] arr;  
 delete[] newArr;  
 return 0;  
}

**Question 6: File I/O Write a program that: Reads numbers from a file "input.txt" (one number per line) Calculates the sum of all even numbers and product of all odd numbers Writes the results to "output.txt" in the format: Sum of even numbers: XX Product of odd numbers: YY Sample input.txt: 3 4 5 2 7 Sample output.txt: Sum of even numbers: 6 Product of odd numbers: 105**

#include <fstream>

using namespace std;

int main() {

ifstream in("input.txt");

ofstream out("output.txt");

int num, sumEven = 0, productOdd = 1;

while (in >> num) {

if (num % 2 == 0) sumEven += num;

else productOdd \*= num;

}

cout << "Sum of even numbers: " << sumEven << endl;

cout << "Product of odd numbers: " << productOdd << endl;

in.close();

out.close();

return 0;

}

5) [3 marks]

// Assume a.txt can be opened successfully and its content is "Welcome to "

#include <iostream>

#include <fstream>

#include <cstring>

int main() {

std::ifstream fin("result.txt");

char str[128];

fin >> str;

strcat(str, " CityU");

std::cout << str;

fin.close();

return 0;

}

ANS: "Welcome CityU"

(6) [3 marks]

#include <iostream>

using namespace std;

class A {

public:

char name = 'A';

};

class B : private A {

public:

B() { name++; }

};

int main() {

B\* b = new B();

cout << b->name;

delete b;

return 0;

}

ANS: Compilation error. "name" is private.

(7) [3 marks]

#include <iostream>

using namespace std;

class A {

public:

float sum(float a, float b) {

return a+b;

}

int sum(int a, int b) {

return a+b;

}

};

int main() {

A \*a = new A();

double x = 0.3, y = 0.4;

cout << a->sum(x, y);

delete a;

return 0;

}

ANS: Compilation error. Ambiguous call to overloaded function.

8) [3 marks]

#include <iostream>

using namespace std;

class Primate {

public:

void sayHi() { cout << "Hi! I'm a primate."; }

};

class Human : public Primate {

public:

void sayHi() { cout << "Hi! I'm a human."; }

};

int main() {

Primate \*human = new Human();

human->sayHi();

delete human;

return 0;

}

ANS: Hi! I'm a primate.

Make this shape if n =3

**1**

**1 2 1**

**1 2 3 2 1**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter height: ";

cin >> n;

for (int row = 1; row <= n; row++) {

// Print leading spaces

for (int i = 0; i < n - row; i++)

cout << " ";

// Print increasing numbers

for (int j = 1; j <= row; j++)

cout << j << " ";

// Print decreasing numbers

for (int j = row - 1; j >= 1; j--)

cout << j << " ";

cout << endl;

}

return 0;

}

**Question 6: File Access with Pointers (20 Marks)**

**The file words.txt contains 1000 lines, each with one English word (e.g., "hello", "world", "test"). Write a program to:**

**Read the words into a dynamic array using pointers (no array indexing like arr[i]).**

**Determine if any word in the file contains exactly 5 characters.**

**Print "Found a 5-letter word" if such a word exists, otherwise print "No 5-letter word found".**

**Properly manage file handling and memory.**

#include <iostream>

#include <fstream>

using namespace std;

bool hasFiveLetterWord(char\*\* words, int size) {

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

char\* ptr = words[i];

int len = 0;

while (\*ptr != '\0') {

len++;

ptr++;

}

if (len == 5)

return true;

}

return false;

}

int main() {

ifstream fin("words.txt");

if (!fin.is\_open()) {

cout << "Unable to open file" << endl;

return -1;

}

// Allocate array to store words (max 1000 lines, 20 chars each)

char\*\* words = new char\*[1000];

for (int i = 0; i < 1000; i++)

words[i] = new char[20];

int count = 0;

char line[20];

while (fin.getline(line, 20) && count < 1000) {

char\* src = line;

char\* dest = words[count];

while (\*src != '\0') {

\*dest = \*src;

src++;

dest++;

}

\*dest = '\0';

count++;

}

fin.close();

// Check for 5-letter word

if (hasFiveLetterWord(words, count))

cout << "Found a 5-letter word" << endl;

else

cout << "No 5-letter word found" << endl;

// Deallocate memory

for (int i = 0; i < 1000; i++)

delete[] words[i];

delete[] words;

return 0;

}

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int n;

cin>>n;

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

{

for (int j = 0; j < i; j++)

{

cout << " ";

}

for(int k = n - i; k > 0; k--){

cout<< "\*";

}

cout << endl;

}

return 0;

}

# # # # #

# # # #

# # #

# # # #

# # # # #

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int n;

cout << "Enter a value of n " << endl;

cin >> n;

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= n; j++)

{

if ((i == 1) || (j == 1) || (i == n) || (j == n) || (i == j) || (i + j - n == 1))

cout << "# ";

else

cout << " ";

}

cout << endl;

}

return 0;

}

**Write a program that includes the following functions and uses 2D arrays. The program should be organized in a functional decomposition manner.**

**int countDiagonals(int arr[][4], int rows): Counts the number of elements on both diagonals of a 4-column 2D array that are greater than 5.**

**void transpose(int arr[][4], int rows): Transposes the 2D array (assuming it’s possible with the given dimensions).**

**void printArray(int arr[][4], int rows): Prints the 2D array.**

**The program should:**

**Declare a 4x4 2D array with initial values {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}}.**

**Print the array.**

**Print the number of elements greater than 5 on the diagonals.**

**Transpose the array and print it again.**

#include <iostream>

using namespace std;

// Count elements > 5 on both diagonals

int countDiagonals(int arr[][4], int rows) {

int count = 0;

for (int i = 0; i < rows && i < 4; i++) {

if (arr[i][i] > 5) count++; // Main diagonal

if (arr[i][3 - i] > 5) count++; // Secondary diagonal

}

return count;

}

// Transpose the 2D array (4x4)

void transpose(int arr[][4], int rows) {

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

for (int j = i + 1; j < 4; j++) {

int temp = arr[i][j];

arr[i][j] = arr[j][i];

arr[j][i] = temp;

}

}

}

// Print the 2D array

void printArray(int arr[][4], int rows) {

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

for (int j = 0; j < 4; j++)

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

cout << endl;

}

}

int main() {

int arr[4][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}};

cout << "Original array:\n";

printArray(arr, 4);

cout << "Number of elements > 5 on diagonals: " << countDiagonals(arr, 4) << endl;

transpose(arr, 4);

cout << "Transposed array:\n";

printArray(arr, 4);

return 0;

}

Question 3: Console Input/Output with Loop (20 Marks)

Write a program that:

* Repeatedly asks the user to input a positive integer n (between 1 and 20).
* Computes the nth Fibonacci number using a loop (not recursion), where F(1) = 1, F(2) = 1, F(n) = F(n-1) + F(n-2) for n > 2.
* Validates the input and continues until the user enters 0 to exit.
* Outputs the result for each valid input.

Example Output:

Enter a positive integer (1-20, 0 to exit): 5

Fibonacci(5) = 5

Enter a positive integer (1-20, 0 to exit): 21

Invalid input! Enter a number between 1 and 20.

Enter a positive integer (1-20, 0 to exit): 0

#include <iostream>

using namespace std;

int main() {

int n;

do {

cout << "Enter a positive integer (1-20, 0 to exit): ";

cin >> n;

if (n == 0) break;

if (n < 1 || n > 20) {

cout << "Invalid input! Enter a number between 1 and 20.\n";

continue;

}

// Compute nth Fibonacci number using a loop

long long fib = 1, prev = 1, temp;

if (n <= 2) {

cout << "Fibonacci(" << n << ") = 1\n";

continue;

}

for (int i = 3; i <= n; i++) {

temp = fib;

fib += prev;

prev = temp;

}

cout << "Fibonacci(" << n << ") = " << fib << endl;

} while (n != 0);

return 0;

}

**Print**

**1**

**1 2 1**

**1 2 3 2 1**

**1 2 1**

**1**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter an odd integer (3-9): ";

cin >> n;

if (n < 3 || n > 9 || n % 2 == 0) {

cout << "Invalid input! Must be an odd integer between 3 and 9.\n";

return 1;

}

int mid = n / 2 + 1;

for (int row = 1; row <= n; row++) {

int spaces = abs(mid - row);

int nums = mid - spaces;

// Print leading spaces

for (int i = 0; i < spaces; i++)

cout << " ";

// Print increasing numbers

for (int j = 1; j <= nums; j++)

cout << j << " ";

// Print decreasing numbers

for (int j = nums - 1; j >= 1; j--)

cout << j << " ";

cout << endl;

}

return 0;

}

Write a program that:

**Dynamically allocates a 2D array of integers based on dimensions rows and cols input by the user (both between 1 and 10).**

**Fills the array with the product of the row and column indices (e.g., element at [i][j] = (i+1)\*(j+1)).**

**Finds the maximum element in the array using pointer arithmetic (no array indexing).**

**Prints the array and the maximum element.**

**Deallocates the array properly.**

Example Output for rows = 2, cols = 3:

Enter rows and columns: 2 3

Array:

1 2 3

2 4 6

Maximum element: 6

#include <iostream>

using namespace std;

int main() {

int rows, cols;

cout << "Enter rows and columns: ";

cin >> rows >> cols;

if (rows < 1 || rows > 10 || cols < 1 || cols > 10) {

cout << "Invalid dimensions! Must be between 1 and 10.\n";

return 1;

}

// Dynamically allocate 2D array

int\*\* arr = new int\*[rows];

for (int i = 0; i < rows; i++)

arr[i] = new int[cols];

// Fill array with product of indices

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

for (int j = 0; j < cols; j++) {

arr[i][j] = (i + 1) \* (j + 1);

}

}

// Find maximum using pointer arithmetic

int max = \*\*arr;

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

int\* rowPtr = arr[i];

for (int j = 0; j < cols; j++) {

if (\*(rowPtr + j) > max)

max = \*(rowPtr + j);

}

}

// Print array

cout << "Array:\n";

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

for (int j = 0; j < cols; j++)

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

cout << endl;

}

cout << "Maximum element: " << max << endl;

// Deallocate array

for (int i = 0; i < rows; i++)

delete[] arr[i];

delete[] arr;

return 0;

}

**Question 6: File Access with Pointers (20 Marks)**

**The file numbers.txt contains 1000 integers, one per line (e.g., "10", "15", "-5"). Write a program to:**

**Read the numbers into a dynamic array using pointers (no array indexing like arr[i]).**

**Determine if there exists a subsequence of exactly 3 consecutive numbers whose sum is divisible by 5.**

**Print "Found a subsequence" if such a subsequence exists, otherwise print "No subsequence found".**

**Use only pointer arithmetic for array access and properly manage file handling and memory.**

Sample numbers.txt:

10

15

-5

20

30

#include <iostream>

#include <fstream>

using namespace std;

bool hasSubsequence(int\* arr, int size) {

if (size < 3) return false;

int\* p = arr;

for (int i = 0; i <= size - 3; i++) {

int sum = \*p + \*(p + 1) + \*(p + 2);

if (sum % 5 == 0) return true;

p++;

}

return false;

}

int main() {

ifstream fin("numbers.txt");

if (!fin.is\_open()) {

cout << "Unable to open file" << endl;

return -1;

}

// Allocate array for up to 1000 integers

int\* arr = new int[1000];

int count = 0;

int num;

while (fin >> num && count < 1000) {

\*(arr + count) = num;

count++;

}

fin.close();

// Check for subsequence

if (hasSubsequence(arr, count))

cout << "Found a subsequence" << endl;

else

cout << "No subsequence found" << endl;

// Deallocate array

delete[] arr;

return 0;

}

**Turtorial 2 (IO TYPE)**

**BMI**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

double height, weight, height\_squared, bmi;

cout << "Enter height (in metre): ";

cin >> height;

cout << "Enter weight (in pound): ";

cin >> weight;

weight = weight / 2.21;

height\_squared = height\*height;

bmi = weight / height\_squared;

cout << fixed << setprecision(2);

cout << "Body Mass Index = " << bmi << endl;

return 0;

**Second to hour / minute/ sec**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int hr, min, sec, inSec;

cout << "Please enter the number of seconds: ";

cin >> inSec;

min = inSec / 60;

sec = inSec % 60;

cout << inSec << " second(s) = "

<< min << " minute(s) "

<< sec << " second(s)" << endl;

hr = min / 60;

min = min % 60;

cout << inSec << " second(s) = "

<< hr << " hour(s) "

<< min << " minute(s) "

<< sec << " second(s)" << endl;

return 0;

}

**Turtorial 3 (If then else )**

**Grade (Socre to A)**

#include <iostream>

using namespace std;

int main(){

char grade;

int mark = 0;

cout << "Input a mark: ";

cin >> mark;

if (mark >= 70) grade = 'A';

else if (mark >= 55) grade = 'B';

else if (mark >= 40) grade = 'C';

else if (mark >= 35) grade = 'D';

else grade = 'F';

cout << "The grade is " << grade;

cout << endl;

}

**Trigangle type (equal , isosceles)**

#include <iostream>

using namespace std;

int main() {

int A,B,C;

cin>>A>>B>>C;

if (A<=0 || B<=0 || C<=0) { cout << "impossible" << endl; }

else if (A+B<=C || A+C<=B || B+C<=A ){ cout << "impossible" << endl;}

else if (A==B && B==C) {cout <<"equilateral" << endl; }

else if (A==B || B==C || C==A) { cout << "isosceles" << endl; }

else { cout << "scalene" << endl;}

return 0;

}

**Taxing problem**

**The program logics can be similar to the following one:**

** Read the net chargeable income. Keep it in a variable (say Income).**

** Declare a variable tax, initialize it to 0.**

** Declare a variable X, initialize it to Income.**

** If X < 40000, then tax = income \* 0.02**

** Else tax = 800 and X = X – 40000**

**If X < 40000, then tax = tax + X \* 0.07**

**Else tax = tax + 2800 and X = X – 40000**

**If X < 40000, then tax = tax + X \* 0.12;**

**Else tax = tax + 4800 and X = X – 40000**

**tax = tax + X \*0.17**

** If tax > Income \* 0.15 then tax = Income \* 0.15**

** Output tax**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double Income;

cout << "Enter net chargeable income: ";

cin >> Income;

double tax = 0;

if (Income <= 40000) {

tax = Income \* 0.02;

}

if (Income > 40000 && Income <= 80000) {

tax = 800;

tax += (Income - 40000) \* 0.07;

}

if (Income > 80000 && Income <= 120000) {

tax = 800;

tax += 2800;

tax += (Income - 80000) \* 0.12;

}

if (Income > 120000) {

tax = 800;

tax += 2800;

tax += 4800;

tax += (Income - 120000) \* 0.17;

}

if (tax > Income \* 0.15) {

tax = Income \* 0.15;

}

cout << fixed << setprecision(2);

cout << "The calculated tax is: $" << tax << endl;

return 0;

}

(Check if the input match x y z )

**Example 1 (chosen characters are x, y, and z)**

**a b**

**The number of inputs matched is 0**

**The number of hidden characters matched is 0**

**Example 2 (chosen characters are x, y, and z)**

**x z**

**The number of inputs matched is 2**

**The number of hidden characters matched is 2**

**Example 3 (chosen characters are z, z, z)**

**z z**

**The number of inputs matched is 2**

**The number of hidden characters matched is 3**

#include <iostream>

using namespace std;

int main() {

char x = 'z', y = 'z', z ='z';

char a, b;

int matched\_input = 0;

int matched\_char = 0;

cin >> a >> b;

if ( a == x || a == y || a == z) matched\_input++;

if ( b == x || b == y || b == z) matched\_input++;

cout << "The number of inputs matched is " <<matched\_input << endl;

if (x == a || x == b) matched\_char++;

if (y == a || y == b) matched\_char++;

if (z == a || z == b) matched\_char++;

cout << "The number of hidden characters matched is " << matched\_char

<< endl;

return 0;

}

**Turtorial 4 (iteration loop)**

**Print Number Matrix (diagonail 0 )**

// print a square matrix with 0 in the diagonal

// and the adjacent cell of the matrix only differ by 1

#include <iostream>

using namespace std;

int main() {

int n, i;

int x, y;

while (1) {

cout << "Enter the number of rows: ";

cin >> n;

if (n <= 0)

cout << "Please input a positive integer." << endl;

if (n > 0) break;

}

for (i = 1; i <= n; i++) {

for (x = 1; x <= n; x++) {

if (x - i == 0) break;

}

// here x contains the value for column 1 at row i

int c;

for (c = 1; c < i; c++) { cout << x - 1 << " "; x--; }

cout << 0 << " ";

y = 1;

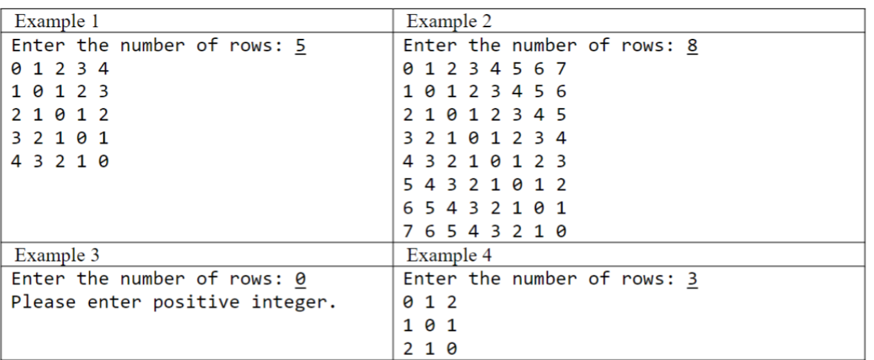
for (c = i + 1; c <= n; c++) { cout << y << " "; y++; }

cout << endl;

}

return 0;

}



**Prime Factorization**

Input a number n: 864

The prime factorization of 864 is:

2 2 2 2 2 3 3 3

Input a number n: 31

The prime factorization of 31 is:

31

// find the prime factorization of n

#include <iostream>

using namespace std;

int main(){

int n; // the number n

bool prime = true;

while (1){

cout << "Input a number n: ";

cin >> n;

if (n >= 1) break;

cout << n << " is not positive." << endl;

}

cout << "The prime factorization of " << n << " is: " << endl;

for (int i = 2; i <= n; i++) {

// if i is a factor of n, repeatedly divide it out

while (n % i == 0) {

cout << i << " ";

n = n / i;

}

}

cout << endl;

return 0;

}

**Input a number n: 1729**

**1729=1^3 + 12^3 = 9^3 + 10^3**

**Input a number n: 100000**

**1729=1^3 + 12^3 = 9^3 + 10^3**

**4104=2^3 + 16^3 = 9^3 + 15^3**

**13832=2^3 + 24^3 = 18^3 + 20^3**

**39312=2^3 + 34^3 = 15^3 + 33^3**

**46683=3^3 + 36^3 = 27^3 + 30^3**

**32832=4^3 + 32^3 = 18^3 + 30^3**

**40033=9^3 + 34^3 = 16^3 + 33^3**

**20683=10^3 + 27^3 = 19^3 + 24^3**

**65728=12^3 + 40^3 = 31^3 + 33^3**

**64232=17^3 + 39^3 = 26^3 + 36^3**

#include <iostream>

using namespace std;

int main() {

int n;

bool prime = true;

while (true) {

cout << "Input a number n: ";

cin >> n;

if (n >= 1) break;

cout << n << " is not positive." << endl;

}

int a, b, c, d;

int a3, b3, c3, d3;

for (a = 1; a <= n; a++) {

a3 = a \* a \* a;

if (a3 > n) break;

for (b = a + 1; b <= n; b++) {

b3 = b \* b \* b;

if (b3 > n) break;

if (a3 + b3 <= n) {

for (c = a + 1; c <= n; c++) {

c3 = c \* c \* c;

if (c3 > n) break;

for (d = c + 1; d <= n; d++) {

d3 = d \* d \* d;

if (d3 > n) break;

if (a3 + b3 == c3 + d3) {

cout << a3 + b3 << "=" << a << "^3 + " << b << "^3 = " << c << "^3 + " << d << "^3" << endl;

}

}

}

}

}

}

return 0;

}

**// estimate the pi number**

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

int main() {

int N, i;

double c, pi;

cin >> N;

c = 0;

for (i = 1; i <= N; i++)

c = c + 1 / (1.0 \* i \* i);

pi = sqrt(c \* 6);

cout << "pi is " << pi << endl;

return 0;

}

**approximates the sine of a given angle x (in radians) using a Taylor series expansion**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

const double pi = 3.141592653589793; // Define pi

double x;

int n;

cin >> x >> n;

if (-pi / 2 <= x && x <= pi / 2) {

double sin\_val; // Avoid using 'sin' (reserved)

double A, B;

A = 0;

B = 0;

for (int i = 1; i <= n; i = i + 4) { // Declare 'i'

double term = 1.0;

for (int j = 1; j <= i; j++) {

term = term \* x / j;

}

A = A + term;

}

for (int i = 3; i <= n; i = i + 4) { // Fixed 'N' to 'n'

double term = 1.0;

for (int j = 1; j <= i; j++) {

term = term \* x / j;

}

B = B + term;

}

sin\_val = A - B;

cout << "sin(" << x << ") = " << sin\_val << endl;

} // Added missing brace

return 0;

}

**Turorial 7 ARRAY**

**Check array equal**

#include <iostream>

using namespace std;

int main()

{

int array1[5] = { 10, 5, 3, 5, 1 };

int array2[5];

int i;

bool arrayEQUAL = true;

cout << "Input 5 numbers\n";

for (i = 0; i < 5; i++)

{

cin >> array2[i];

}

for (i = 0; i < 5 && arrayEQUAL; i++)

{

if (array1[i] != array2[i])

{

arrayEQUAL = false;

}

}

if (arrayEQUAL)

{

cout << "The arrays are equal";

}

else

{

cout << "The arrays are not equal";

}

return 0;

}

**Finding number in array**

#include <iostream>

using namespace std;

int linearSearch(int arr[], int size, int target) {

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

if (arr[i] == target) {

return i; // Return index if found

}

}

return -1; // Return -1 if not found

}

int main() {

int arr[] = {5, 2, 9, 1, 5, 6};

int size = sizeof(arr) / sizeof(arr[0]);

int target = 9;

int result = linearSearch(arr, size, target);

if (result != -1) {

cout << "Element found at index: " << result << endl;

} else {

cout << "Element not found" << endl;

}

return 0;

}

#include <iostream>

using namespace std;

int binarySearch(int arr[], int size, int target) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) {

return mid;

}

if (arr[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

int main() {

int arr[] = {1, 2, 3, 4, 5, 6}; // Must be sorted

int size = sizeof(arr) / sizeof(arr[0]);

int target = 4;

int result = binarySearch(arr, size, target);

if (result != -1) {

cout << "Element found at index: " << result << endl;

} else {

cout << "Element not found" << endl;

}

return 0;

}

#include <iostream>

using namespace std;

int main()

{

const int N = 5;

int a[N], i, x, pos;

for (i = 0; i < N; i++)

{

cout << "Enter element " << i << ": ";

cin >> a[i];

}

cout << "Input your target: ";

cin >> x;

for (i = 0, pos = -1; i < N; i++)

{

if (a[i] == x)

{

pos = i;

break;

}

}

if (pos == -1)

cout << "Target not found!\n";

else

cout << "Target found at position " << pos << endl;

return 0;

}

**Ask the user to input an integer i where 1  i  9.**

** For each element in data[], print a bar of ‘\*’**

**where the number of ‘\*’ in the bar is the same as the**

**value in that array element.**

** However, for the bar where the position of data[]**

**is same as the integer inputted by the user, print a**

**bar of ‘+’ instead.**

** sample output**

**Input a digit: 10**

**Input a digit: 0**

**Input a digit: 2**

**\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\***

**++**

**\*\*\*\*\***

**\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\***

**\***

**\*\*\*\*\***

#include <iostream>

using namespace std;

int main() {

int data[10] = { 5, 10, 2, 5, 8, 8, 7, 9, 1, 5 };

int i = 0; int h = 0;

while (i < 1 || i > 9) {

cout << "Input a digit: ";

cin >> i;

}

for (int h = 0; h <= 9; h++){

if (i == h){

for (int k = 0; k < data[h]; k++)

cout << '+';

cout << endl;

}

else {

for (int k = 0; k < data[h]; k++)

cout << '\*';

cout << endl;

}

} // end-for

return 0;

}

**Rolling 2 dices**

**1 occurrence(s) of the sum 2**

**2 occurrence(s) of the sum 3**

**3 occurrence(s) of the sum 4**

**4 occurrence(s) of the sum 5**

**5 occurrence(s) of the sum 6**

**6 occurrence(s) of the sum 7**

**5 occurrence(s) of the sum 8**

**4 occurrence(s) of the sum 9**

**3 occurrence(s) of the sum 10**

**2 occurrence(s) of the sum 11**

**1 occurrence(s) of the sum 12**

#include <iostream>

using namespace std;

int main(){

int sum[12] = {0}; // max 12 values

for (int i = 1; i <= 6; i++) // i is dice 1

for (int j = 1; j <= 6; j++) // j is dice 2

// encounter the sum i+j,

// increment the count kept in the array sum[].

// note: the first element of sum is sum[0]

sum[i+j-1] = sum[i+j-1] + 1;

for (int k = 0; k < 12; k++)

if (sum[k] != 0)

{

cout << sum[k]

<< " occurrence(s) of the sum "

<< k+1 << endl;

}

}

**Turtorial 8 String**

strcpy(dest, src) Copy the content of string src to the string dest

strcat(dest, src) Append the content of string src onto the end of string dest

strcmp(s1, s2) Lexicographically compare two strings, s1 and s2, character by character. 3 possible return value:

• 0: s1 and s2 are identical

• >0: s1 is greater than s2

• <0: s1 is less than s1

strlen(s1) Returns the length, which is defined as the number of characters (exclude the null character) containing in the string s1

**Q1 reverses a word (with less than 100 characters)**

**inputted from Console, and**

**2. counts the number of characters inputted.**

**Example :**

**qwertyuiop**

**No. of characters = 10**

**poiuytrewq**

#include <iostream>

using namespace std;

int main(){

char c[100];

int i;

cin.getline(c,100);

for(i=0; c[i]!='\0'; i++);

cout << "No. of characters = " << i << endl;

for(int j=i-1; j>=0; j--)

cout << c[j];

cout << endl;}

**Comparison**

**Example :**

**hello**

**world**

**No. of characters at common positions: 0**

**transparent**

**teenroom**

**No. of characters at common positions: 2**

#include <iostream>

using namespace std;

int main(){

char w1[100];

char w2[100];

char w3[100];

int i, count;

cin >> w1;

cin >> w2;

for(i=0; w1[i]!='\0'; i++);

for(int j=i-1; j>=0; j--)

w3[i - j - 1] = w1[j];

w3[i] = '\0';

i = 0; count = 0;

while (w3[i] != '\0' && w2[i] != '\0'){

if (w3[i] == w2[i])

count++;

i++;

}

cout << "No. of characters at common positions: "

<< count << endl;

}

**String conversion**

**123456789**

**d10 is X**

**000000009**

**d10 is 4**

#include <iostream>

#include <cstdlib>

using namespace std;

int main() {

char isbn[11]; // one more place for safety;

char w[2];

char d10;

int i, sum = 0;

for (i = 0; i < 9; i++)

cin >> isbn[i];

// cin >> isbn; // or using cstring with cin directly

isbn[i] = '\0';

w[1] = '\0';

i = 8;

while (i >= 0) { // decrement from 8 to 0

w[0] = isbn[i];

sum = sum + (10 - i) \* atoi(w);

i--;

}

sum = sum % 11;

if (sum == 0)

isbn[9] = 0;

else

isbn[9] = 11 - sum;

if (isbn[9] == 10)

d10 = 'X';

else

d10 = '0' + isbn[9];

cout << "d10 is " << d10 << endl;

}

Turtorial 11(recursion )

**von Neumann ordinal**

**int main() {**

**vonNeumann(0); cout << endl;**

**vonNeumann(1); cout << endl;**

**vonNeumann(2); cout << endl;**

**vonNeumann(3); cout << endl;**

**vonNeumann(4); cout << endl;**

**}**

**{}**

**{{}}**

**{{}, {{}}}**

**{{}, {{}}, {{}, {{}}}}**

**{{}, {{}}, {{}, {{}}}, {{}, {{}}, {{}, {{}}}}}**

#include <iostream>

using namespace std;

void vonNeumann(int N) {

if (N == 0)

{ cout << "{}";

return;

}

cout << "{";

for (int i = 0; i < N - 1; i++){

vonNeumann(i);

cout << ", ";

}

vonNeumann(N-1);

cout << "}";

return;

}

int main() {

vonNeumann(0); cout << endl;

vonNeumann(1); cout << endl;

vonNeumann(2); cout << endl;

vonNeumann(3); cout << endl;

vonNeumann(4); cout << endl;

**Input a value [1..50]:**

**9**

**0: 0**

**1: 1**

**2: 1**

**3: 2**

**4: 3**

**5: 5**

**6: 8**

**7: 13**

**8: 21**

**9: 34**

**The sum of the first 9**

**Fibonacci numbers is 88**

#include <iostream>

using namespace std;

int fib(int n) {

// base case

if (n == 0) return 0; // this base case can be skipped.

if (n == 1) return 1;

if (n == 2) return 1;

// n is odd

if (n % 2 != 0) {

int a = fib((n+1)/2);

int b = fib((n-1)/2);

return a\*a + b\*b;

}

// n is even

int a = fib(n/2 + 1);

int b = fib(n/2 - 1);

return a\*a - b\*b;

}

void main() {

int N;

int sum = 0;

int number = 0;

cout << "Input a value [1..50]:\n";

cin >> N;

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

number = fib(i);

sum += number;

cout << i << ": " << number << endl;

}

cout << "The sum of the first " << N << " Fibonacci numbers is " << sum << endl;

}

**Array searching**

#include <iostream>

#include <iomanip>

using namespace std;

char days[7][10] = { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",

"Saturday", "Sunday" };

char soln[7][10] = { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",

"Saturday", "Sunday" };

void up(char c1, int x, int y);

void down(char c1, int x, int y);

void right(char c1, int x, int y);

void left(char c1, int x, int y);

int main() {

for (int i = 0; i < 7; i++) {

for (int j = 0; days[i][j] != 0; j++)

cout << \*(\*(days + i) + j); // cout << days[i][j];

cout << endl;

}

cout << "Input xand y: ";

cin >> x >> y;

soln[x][y] = '\*’;

int c1 = days[x][y];

up(c1, x + 1, y);

down(c1, x - 1, y);

left(c1, x, y - 1);

right(c1, x, y + 1);

cout << "The cells reachable from ("

<< x << "," << y << ") are masked:" << endl;

for (int k = 0; k < 7; k++)

cout << soln[k] << endl;

} void up(char c1, int x, int y) {

if (x >= 0 && x <= 6) {

// cell(x,y) is valid

if (days[x][y] > c1 && soln[x][y] != '\*') {

soln[x][y] = '\*’;

int c2 = days[x][y];

up(c2, x + 1, y);

down(c2, x - 1, y);

left(c2, x, y - 1);

right(c2, x, y + 1);

}

}

}

void down(char c1, int x, int y) { up(c1, x, y); }

void right(char c1, int x, int y) {

if (y >= 0 && y <= 9) {

// cell(x,y) is valid

if (days[x][y] > c1 && soln[x][y] != '\*') {

soln[x][y] = '\*’;

int c2 = days[x][y];

up(c2, x + 1, y);

down(c2, x - 1, y);

left(c2, x, y - 1);

right(c2, x, y + 1);

}

}

}

void left(char c1, int x, int y) { right(c1, x, y); }

Tutorial 12 (Class and object )

#include <iostream>

using namespace std;

class myDoor {

protected:

bool state;

public:

myDoor(int s) { if (s == 1) state = true; else state = false; }

void close() { state = false; }

void open() { state = true; }

void change() { if (state == true) close(); else open(); }

void print() { if (state == true) cout << "Open" << endl; else cout << "Closed" << endl; }

bool isEqual(myDoor door2) { if (door2.state == state) return true; else return false; }

};

int main() {

int i, n;

cout << "What is the initial state of your door (1 for open, and 0 for closed): ";

do { cin >> i; } while (i != 1 && i != 0);

myDoor o2(i);

cout << "What is the number of times you want to change the state of your door: ";

do { cin >> n; } while (n < 0);

for (int j = 0; j < n; j++) o2.change();

o2.print();

myDoor o3(1), o4(0);

if (o2.isEqual(o3)) { o2.print(); cout << " same as o3" << endl; }

if (o2.isEqual(o4)) { o2.print(); cout << " same as o4" << endl; }

}

char a = 'a', b = 'e';

if (a++ == 'b')

cout << a;

if (--b == 'd')

cout << b;

a += b-a;

cout << a;

• Compare first assign later for (a++ == ‘b’),not executed

• Assign first compare later for (--b == ‘d’), executed. Output d.

• Now a = ‘b’, b = ‘d’. b-a=‘d’-’b’=2

• a=a+2=‘d’. Output d

Correct Answer:

dd

int a = 1;

for (int i = 50; i>0; i--) {

if (i % 5 == 0)

a = a+ 1;

a = a - 1;

}

cout << a;

• For loop executes 50 times, (i % 5 == 0) satisfied 10 times,

• a=1-50+10=-39

Correct Answer:

-39

int j(0);

char c = 'a', d = 'z';

// ascii code for 'a' and 'z' are 97 and 122, respectively

while (d % c) {

for (int i = 1 ; i <= (d %c); i++)

switch (i%2) {

case 0: j++; break;

case 1: break;

}

if (j > 20) break;

c++;

}

cout << j;

• break is used to jump out of the switch statement

• i=1, d%c=25, i%2=1,

• i=2, d%c=25, i%2=0, j++,

• ...

• i=25, d%c=25, i%2=1, j++,

• j=12 now

• Then since j<20, c=‘b’(98), d%c=24

• The similar process...j=24 now

• Since j>20, break, output the value of j which is 24

Correct Answer:

24

**Mock final exam**

**Q1**

double x = 5.7;

int y = x / 2; // y = 5.7 / 2 = 2 (integer division after type conversion)

double z = x + y \* 2; // z = 5.7 + 2 \* 2 = 5.7 + 4 = 9.7

cout << z; // Outputs: 9.7

**Q2**

int a = 3, b = 3;

if (a == b) // True (3 == 3)

if (a > 4) // False (3 > 4)

cout << a - b;

else // Executes due to false condition in inner if

cout << b + 1; // Outputs: 3 + 1 = 4

cout << a; // Outputs: 3

// Final output: 43

**Q3**

int x = 5, y = 2, z = 1;

if (z = 0) { // Assignment, sets z to 0, condition evaluates to 0 (false)

if (y == 2) cout << x;

else cout << y;

}

cout << z; // Outputs: 0

// Since if condition is false, only z is printed

**Q4’**

int a = 1, b = 5, c = 5;

bool result = a++ < b || b > 0 && c > a-- || ++c < b;

// Step 1: a++ < b → 1 < 5 (true), a becomes 2

// Short-circuit: Since the first condition is true, the rest of the expression is not evaluated

// a = 2 (from a++), b = 5, c = 5 (no change), result = true (1)

cout << a << " " << b << " " << c << " " << result;

// Outputs: 2 5 5 1

**Q5**

void increment(int\* p) {

**p = p + 1;**  // This only increments the local pointer p, not the value it points to

}

int main() {

int x = 4;

int\* ptr = &x;

increment(ptr);

cout << x; // Outputs: 4 (unchanged)

}

// To increment x to 5, modify the function to dereference p and increment the value

Corrected Statement: In the increment function, **change p = p + 1; to \*p = \*p + 1;**

**Q6**

**void printPattern(int n) {**

**if (n <= 0)**

**cout << "\*" << endl;**

**else {**

**printPattern(n - 1);**

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

**cout << "\*";**

**cout << endl;**

**}**

**}**

**// printPattern(3):**

**// n = 3 → calls printPattern(2)**

**// n = 2 → calls printPattern(1)**

**// n = 1 → calls printPattern(0)**

**// n = 0 → prints "\*\n"**

**// n = 1 → prints "\*\*\n"**

**// n = 2 → prints "\*\*\*\n"**

**// n = 3 → prints "\*\*\*\*\n"**

**// Final output:**

**// \***

**// \*\***

**// \*\*\***

**// \*\*\*\***

Q7

int\* arr = new int[4];

for (int i = 0; i < 4; i++) {

arr[i] = i + 1; // arr = {1, 2, 3, 4}

}

int\* p = arr;

\*(p + 1) = \*(p) + 1; // arr[1] = arr[0] + 1 = 1 + 1 = 2 (unchanged)

\*(p + 2) = \*(p + 1) + 1; // arr[2] = arr[1] + 1 = 2 + 1 = 3 (unchanged)

\*(p + 3) = \*(p + 2); // arr[3] = arr[2] = 3

// arr = {1, 2, 3, 3}

for (int i = 0; i < 4; i++) {

cout << arr[i] << " "; // Outputs: 1 2 3 3

}

**Q8**

void swapValues(int m, int n) {

int temp = m + 1;

m = n + 1;

n = temp + 1;

}

int main() {

int a = 6, b = 3;

swapValues(a, b);

cout << a << " " << b; // Outputs: 6 3 (unchanged due to pass by value)

}

// Desired output: 4 7

// To swap, need pass by reference or pointers. Since only one statement can be changed, modify the function signature

**Corrected Statement: Change void swapValues(int m, int n) to void swapValues(int& m, int& n)**

**Question 2 (15%) – Code Development with Functional Decomposition**

**Instructions: Write a program that uses functional decomposition to process an array of integers. The program should include at least two functions (besides main) to compute the sum of even numbers and the product of odd numbers in the array. Use the following array in main: int arr[5] = {1, 2, 3, 4, 5};. Output the sum of even numbers and the product of odd numbers in the format:**

**Sum of even numbers: <sum>**

**Product of odd numbers: <product>**

**Sample Output:**

**Sum of even numbers: 6**

**Product of odd numbers: 15**

#include <iostream>

using namespace std;

// Function to compute sum of even numbers

int sumEven(int arr[], int size) {

int sum = 0;

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

if (arr[i] % 2 == 0) {

sum += arr[i];

}

}

return sum;

}

// Function to compute product of odd numbers

int productOdd(int arr[], int size) {

int product = 1;

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

if (arr[i] % 2 != 0) {

product \*= arr[i];

}

}

return product;

}

int main() {

int arr[5] = {1, 2, 3, 4, 5};

int evenSum = sumEven(arr, 5);

int oddProduct = productOdd(arr, 5);

cout << "Sum of even numbers: " << evenSum << endl;

cout << "Product of odd numbers: " << oddProduct << endl;

return 0;

}

**Q3 (adding odd number as input and igore even number and avg)**

**Enter an integer (negative to stop): 2**

**Enter an integer (negative to stop): 4**

**Enter an integer (negative to stop): -1**

**No odd numbers entered.**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int num;

int sum = 0, count = 0;

cout << "Enter an integer (negative to stop): ";

cin >> num;

while (num >= 0) {

if (num % 2 != 0 && num > 0) { // Check for positive odd numbers

sum += num;

count++;

}

cout << "Enter an integer (negative to stop): ";

cin >> num;

}

if (count == 0) {

cout << "No odd numbers entered." << endl;

} else {

double avg = static\_cast<double>(sum) / count;

cout << "Average of odd numbers: " << fixed << setprecision(2) << avg << endl;

}

return 0;

}

**Shape printing**

#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter N (1-9): ";

cin >> N;

if (N < 1 || N > 9) {

cout << "Invalid input!" << endl;

return 0;

}

int totalRows = 2 \* N - 1;

int midRow = N;

for (int row = 1; row <= totalRows; row++) {

int spaces = abs(midRow - row);

int symbols = 2 \* (N - spaces) - 1;

// Print leading spaces

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

cout << " ";

}

// Print symbols

if (symbols == 1) {

cout << "\*";

} else {

cout << "\*";

for (int i = 0; i < symbols - 2; i++) {

cout << "+";

}

cout << "\*";

}

cout << endl;

}

return 0;

}

**Q5**

**Enter the size of the array: 4**

**Enter 4 integers: 5 2 8 1**

**Difference between max and min: 7**

#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter the size of the array: ";

cin >> N;

if (N < 1 || N > 100) {

cout << "Invalid size!" << endl;

return 0;

}

int\* arr = new int[N];

cout << "Enter " << N << " integers: ";

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

cin >> arr[i];

}

int maxVal = arr[0], minVal = arr[0];

for (int i = 1; i < N; i++) {

if (arr[i] > maxVal) maxVal = arr[i];

if (arr[i] < minVal) minVal = arr[i];

}

cout << "Difference between max and min: " << (maxVal - minVal) << endl;

delete[] arr; // Deallocate memory

return 0;

}

**Q6**

**Data txt :**

**10**

**15**

**5**

**20**

**8**

**Sample Output:**

**Enter the threshold: 12**

**Sum of integers greater than threshold: 35**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream inFile("data.txt");

if (!inFile) {

cout << "Error opening file!" << endl;

return 1;

}

// Count number of integers in the file

int count = 0, temp;

while (inFile >> temp) {

count++;

}

inFile.clear();

inFile.seekg(0); // Reset file pointer to beginning

// Dynamically allocate array

int\* arr = new int[count];

int\* ptr = arr;

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

inFile >> \*(ptr + i); // Use pointer to fill array

}

inFile.close();

// Get threshold and compute sum

int threshold;

cout << "Enter the threshold: ";

cin >> threshold;

int sum = 0;

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

if (\*(ptr + i) > threshold) {

sum += \*(ptr + i);

}

}

cout << "Sum of integers greater than threshold: " << sum << endl;

delete[] arr; // Deallocate memory

return 0;

}

**Mock exam final 2**

(a)

#include <iostream>

using namespace std;

int main() {

int x = 3, y = 4;

cout << x++ << " " << ++y << endl;

return 0;

}

Answer:

3 5

(b)

#include <iostream>

using namespace std;

int main() {

int a = 5;

if (a = 2)

cout << "Yes" << endl;

else

cout << "No" << endl;

return 0;

}

ANS YES

**(d)**

#include <iostream>

using namespace std;

int main() {

int x = 10;

int \*p = &x;

\*p = \*p + 2;

cout << x << endl;

return 0;

}

Answer:

12

(e)

#include <iostream>

using namespace std;

void modify(int &n) {

n = n \* 2;

}

int main() {

int a = 3;

modify(a);

cout << a << endl;

return 0;

}

Answer:

6

modify doubles a (passed by reference) from 3 to 6.

(f)

#include <iostream>

using namespace std;

int main() {

char str[] = "Test";

char \*p = str;

cout << \*p++ << endl;

return 0;

}

Answer:

T

\*p++ outputs the first character ('T') and increments the pointer.

(g)

#include <iostream>

using namespace std;

int main() {

int arr[2][2] = {{1, 2}, {3, 4}};

cout << arr[1][0] << endl;

return 0;

}

Answer:

3

(h)

#include <iostream>

using namespace std;

int factorial(int n) {

if (n <= 1) return 1;

return n \* factorial(n - 1);

}

int main() {

cout << factorial(4) << endl;

return 0;

}

(i)

#include <iostream>

using namespace std;

int main() {

int \*p = new int[3];

p[0] = 1; p[1] = 2; p[2] = 3;

cout << p[1] << endl;

delete[] p;

return 0;

**(j)**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream fout("test.txt");

fout << "Hello";

fout.close();

return 0;

}

**Answer:No console output. The program writes "Hello" to "test.txt".**

**Question 4 [20 marks] - Shape Printing**

**Write a C++ program that takes an integer N from the user and prints a diamond shape with**

**2N-1 rows using asterisks (\*).**

**Sample Input/Output:For N = 3:**

**\***

**\* \***

**\* \***

**\* \***

**\***

**Answer:**

#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter N: ";

cin >> N;

int rows = 2 \* N - 1;

for (int i = 1; i <= N; i++) {

for (int j = 1; j <= N - i; j++) cout << " ";

cout << "\*";

for (int j = 1; j < 2 \* (i - 1); j++) cout << " ";

if (i > 1) cout << "\*";

cout << endl;

}

for (int i = N - 1; i >= 1; i--) {

for (int j = 1; j <= N - i; j++) cout << " ";

cout << "\*";

for (int j = 1; j < 2 \* (i - 1); j++) cout << " ";

if (i > 1) cout << "\*";

cout << endl;

}

return 0;

}

**Question 5 [20 marks] - Dynamic Array**

**Write a C++ program that dynamically allocates an array based on a user-specified size,**

**reads elements, and outputs the sum of elements at odd indices (1, 3, etc.).**

**Sample Input/Output:**

**Enter array size: 4**

**Enter element 0: 1**

**Enter element 1: 2**

**Enter element 2: 3**

**Enter element 3: 4**

**Sum at odd indices: 6**

**Answer:**

#include <iostream>

using namespace std;

int main() {

int size;

cout << "Enter array size: ";

cin >> size;

int \*arr = new int[size];

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

cout << "Enter element " << i << ": ";

cin >> arr[i];

}

int sum = 0;

for (int i = 1; i < size; i += 2) {

sum += arr[i];

}

cout << "Sum at odd indices: " << sum << endl;

delete[] arr;

return 0;

}

**Odd indices (1, 3) have values 2 and 4, sum is 6.**

**Question 6 [20 marks] - Program with File I/O**

**Write a C++ program that reads integers from "numbers.txt", counts how many are even, and**

**writes the count to "result.txt".**

**Sample Input (numbers.txt):**

**3 8 2 7 4**

**Sample Output (result.txt):**

**3**

**Answer:**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream fin("numbers.txt");

ofstream fout("result.txt");

if (!fin.is\_open() || !fout.is\_open()) {

cout << "File error." << endl;

return 1;

}

int num, count = 0;

while (fin >> num) {

if (num % 2 == 0) count++;

}

fout << count << endl;

fin.close();

fout.close();

return 0;

}

**Even numbers (8, 2, 4) result in a count of 3.**

**Mock Final exam 3**

**Q1**

int a = 3, b = 5;

if (a == 3)

b = b + 2;

cout << b;

Output:

The condition a == 3 is true, so b will be updated to 5 + 2 = 7.

The output will be 7.

**Q2**

Fix the Bug

int x = 3;

if (x = 4) // x==4

cout << "Correct";

else

cout << "Wrong";

**Q3**

int x = 0, y = 1;

if (x && y)

cout << "True";

else

cout << "False";

Output:

The condition x && y uses short-circuiting. Since x is 0 (false), the second part of the condition is not evaluated, and it directly goes to the else block, printing "False".

**Q4**

int arr[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) { // ERROR: Should be j < 3

cout << arr[i][j];

}

}

**Q2 This question will focus on writing a function that decomposes a problem into smaller, manageable functions. Example problem:**

**Task: Write a C++ program that accepts a purchase amount and calculates the appropriate discount according to the following rules:**

**If the purchase is between $1,000 and $2,999, apply a 10% discount.**

**For amounts between $3,000 and $4,999, apply a 20% discount on the amount exceeding $2,999.**

**If the purchase exceeds $5,000, apply a 30% discount on the amount over $4,999.**

**Use functional decomposition to structure your code into helper functions like calculate\_discount, apply\_discount, etc. (10 points)**

#include <iostream>

using namespace std;

double calculate\_discount(double purchase) {

if (purchase >= 10000) {

return purchase \* 0.25;

} else if (purchase >= 5000) {

return (purchase - 4999) \* 0.30 + 2999 \* 0.20 + 2999 \* 0.10;

} else if (purchase >= 3000) {

return (purchase - 2999) \* 0.20 + 2999 \* 0.10;

} else if (purchase >= 1000) {

return purchase \* 0.10;

} else {

return 0;

}

}

int main() {

double purchase;

cout << "Enter the purchase amount: ";

cin >> purchase;

double discount = calculate\_discount(purchase);

cout << "The discount is: " << discount << endl;

return 0;

}

**Q4 (Shape Printing)**

**Enter number of rows: 5**

**1**

**2 3 2**

**3 4 5 4 3**

**2 3 4 3 2**

**1**

#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter number of rows: ";

cin >> N;

for (int row = 1; row <= N; row++) {

// Print leading spaces

for (int space = 1; space <= N - row; space++) {

cout << " ";

}

// Print ascending numbers

for (int num = row; num < row \* 2; num++) {

cout << num << " ";

}

// Print descending numbers

for (int num = row \* 2 - 2; num >= row; num--) {

cout << num << " ";

}

cout << endl;

}

return 0;

}

**Q5(Doubling the element of array )**

#include <iostream>

using namespace std;

int main() {

int N;

cout << "Enter number of elements: ";

cin >> N;

int\* arr = new int[N];

cout << "Enter elements: ";

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

cin >> arr[i];

}

cout << "Doubled elements: ";

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

arr[i] \*= 2;

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

}

cout << endl;

delete[] arr; // Free dynamically allocated memory

return 0;

}

Mock exam final (4)

Q1

#include <iostream>

using namespace std;

int main() {

int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

int\* p = (s + 9 / 3);

cout << \*p << " ";

p = p + s[p[2] / 9]; // Error here

cout << \*p;

return 0;

}

**//Expected Output: 4 6**

**Correction: p = p + s[9 / p[2]]; (to get p pointing to 6).**

**Q2**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

char line[60]; ifstream f; int count = 0;

f.open("input.txt");

if (f.is\_open()) {

while (f.eof()) { // Error here

f.getline(line, 60);

count++;

}

}

cout << count;

f.close();

return 0;

}

**Correction: while (!f.eof()) (loop until end of file).**

**Q3**

**Expected Output:**

**\***

**\*\*\***

**\*\*\*\*\***

#include <iostream>

using namespace std;

void print(int size) {

if (size <= 0) { // Error here

cout << "" << endl;

} else {

print(size - 2);

for (int i = 0; i < size; i++) cout << "\*";

cout << endl;

}

}

int main() {

print(5);

return 0;

}

**Correction: if (size <= 1) (to avoid extra \*).**

**Q4**

**Expected Output: 1 3 2**

#include <iostream>

using namespace std;

int main() {

int\* s = new int[3];

for (int i = 0; i < 3; i++) s[i] = i;

int\* p = s;

\*(p) = \*(p) + 1;

\*(p + 1) = \*(p) + 2; // Error here

\*(p + 2) = \*p;

for (int i = 0; i < 3; i++) cout << s[i] << " ";

delete[] s; // Must free memory

return 0;

}

**Correction: \*(p + 1) = \*(p + 2); (to correctly set s[1] = 2).**

**Q2**

**Task: Write a program that: Reads an array of integers.**

**Implements three functions:**

**findMinMax() (returns min and max via pointers).**

**sortArray() (ascending order).**

**searchValue() (returns index if found, else -1).**

#include <iostream>

using namespace std;

void findMinMax(int\* arr, int size, int\* min, int\* max) {

\*min = \*max = arr[0];

for (int i = 1; i < size; i++) {

if (arr[i] < \*min) \*min = arr[i];

if (arr[i] > \*max) \*max = arr[i];

}

}

void sortArray(int\* arr, int size) {

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

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

if (arr[j] > arr[j+1]) swap(arr[j], arr[j+1]);

}

int searchValue(int\* arr, int size, int key) {

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

if (arr[i] == key) return i;

return -1;

}

int main() {

int arr[] = {5, 3, 9, 1, 7};

int size = 5, min, max, key = 9;

findMinMax(arr, size, &min, &max);

sortArray(arr, size);

int index = searchValue(arr, size, key);

cout << "Min: " << min << ", Max: " << max << endl;

cout << "Sorted Array: ";

for (int i = 0; i < size; i++) cout << arr[i] << " ";

cout << "\nKey " << key << " found at index " << index;

return 0;

}

**Question 3: Console I/O Program (15 Marks)**

**Task: Write a menu-driven program that:**

**Asks for numbers until -1 is entered.**

**Computes:**

**Sum of even numbers.**

**Product of odd numbers.**

**Validates input (only integers allowed).**

#include <iostream>

using namespace std;

int main() {

int num, sumEven = 0, productOdd = 1;

cout << "Enter numbers (-1 to stop): ";

while (cin >> num && num != -1) {

if (num % 2 == 0) sumEven += num;

else productOdd \*= num;

}

cout << "Sum of evens: " << sumEven << endl;

cout << "Product of odds: " << productOdd << endl;

return 0;

}

**Q4**

**1**

**232**

**34543**

**4567654**

**567898765**

**Task: Print a right-aligned number pyramid for input N=5:**

#include <iostream>

using namespace std;

int main() {

int N = 5;

for (int i = 1; i <= N; i++) {

for (int j = 1; j <= N - i; j++) cout << " ";

for (int j = i; j <= 2\*i-1; j++) cout << j;

for (int j = 2\*i-2; j >= i; j--) cout << j;

cout << endl;

}

return 0;

}

**Q5**

**Question 5: Dynamic Arrays (15 Marks)**

**Task:**

**Dynamically allocate an array of size N (user input).**

**Fill it with random numbers (1-100).**

**Resize the array to N+5 and append 5 new random numbers.**

**Print before & after resizing.**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main() {

srand(time(0));

int N;

cout << "Enter size: ";

cin >> N;

int\* arr = new int[N];

for (int i = 0; i < N; i++) arr[i] = rand() % 100 + 1;

int\* newArr = new int[N + 5];

for (int i = 0; i < N; i++) newArr[i] = arr[i];

for (int i = N; i < N + 5; i++) newArr[i] = rand() % 100 + 1;

delete[] arr;

arr = newArr;

N += 5;

cout << "Resized array: ";

for (int i = 0; i < N; i++) cout << arr[i] << " ";

delete[] arr;

return 0;

}

**Q6**

**Read words.txt (one word per line).**

**Store words in a dynamic 2D array (char\*\*).**

**Find and print all words starting with user-input prefix (using only pointer arithmetic).**

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

int main() {

ifstream file("words.txt");

if (!file) { cerr << "File error!"; return 1; }

char buffer[100];

char\*\* words = new char\*[1000]; // Assume max 1000 words

int count = 0;

while (file.getline(buffer, 100)) {

words[count] = new char[strlen(buffer) + 1];

strcpy(words[count], buffer);

count++;

}

char prefix[100];

cout << "Enter prefix: ";

cin >> prefix;

bool found = false;

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

if (strncmp(words[i], prefix, strlen(prefix)) == 0) {

cout << prefix << " is a prefix of " << words[i] << endl;

found = true;

}

}

if (!found) cout << prefix << " is not in the file." << endl;

for (int i = 0; i < count; i++) delete[] words[i];

delete[] words;

return 0;

}