**XII Computer Science**

**Practical File**

**Name :** Varad Mahashabde

**Class :** XII **Sec :** A

**Roll no.:** 29

1. Guessing Game

#include <iostream>

#include <iomanip>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

int getDigitSum(int x, int base) {

int sum = 0;

while (x > 0) {

sum += x % base;

x /= base;

}

return sum;

}

int main () {

time\_t k;

srand( unsigned(time(&k)) );

// cout << (unsigned int)(-1) << "\n\n";

// for (int i = 0; i < 255; ++i)

// cout << rand() << endl;

int guess, luckys[3];

bool won = false;

do {

cout << "Enter your guess : "; cin >> guess;

if (guess < 2 || guess > 19)

cout << "ERR : INPUT OUT OF BOUNDS (2 - 19)\n";

} while (guess < 2 || guess > 19);

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

luckys[i] = (199 - 101) \* float(rand()) / 0x7fff + 101;

cout << "The lucky numbers are : \n\t";

for (int i = 0 ; i < 3; ++i) cout << setw(8) << luckys[i];

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

luckys[i] = getDigitSum(luckys[i], 10);

won |= luckys[i] == guess;

}

cout << "\nThe digit sums are : \n\t";

for (int i = 0 ; i < 3; ++i) cout << setw(8) << luckys[i];

if (won)

cout << "\nYES YOU ARE THE WINNERRRRRR!!!!\nnow get out\n";

else

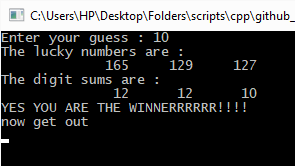
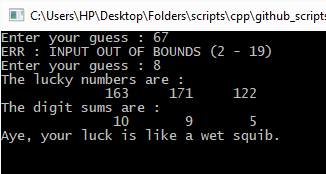
cout << "\nAye, your luck is like a wet squib.";

while (getchar() != '\n');

getchar();

}

Output :



1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

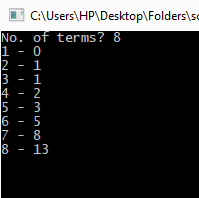
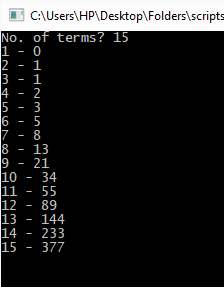
cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Series

#include <iostream>

#include <stdio.h>

#include <iomanip>

using namespace std;

long fact(int n) {

long res = 1;

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

res \*= i;

cout << res;

}

long double pow(long double b ,int e) {

long double res = 1;

if (e < 0)

for (int i = -e; i > 0; ++i)

res /= b;

else if (e > 0)

for (int i = e; i > 0; ++i)

res /= b;

return res;

}

int main() {

long double x, sum = 0; unsigned int n;

cout << "Give value of x : "; cin >> x;

cout << "No. of terms? "; cin >> n;

for (int i = 2\*n - 1; i > 1; i -= 2)

sum += (((i - 1)/2) % 2)?(-1):(1) \* pow(x, i) / fact(i);

sum += 1;

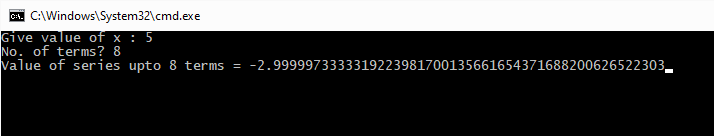
cout << "Value of series upto " << n << " terms = " << setprecision(50) << sum;

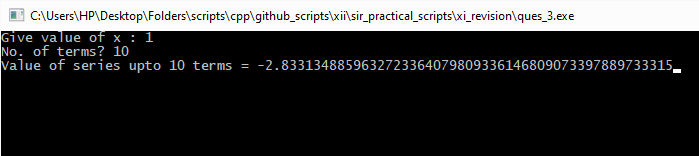
while (getchar() != '\n');

getchar();

}

Output :





1. Triangles

#include <iostream>

#include <stdio.h>

using namespace std;

int main () {

unsigned int lines, choice;

cout << "Enter no. of lines : "; cin >> lines;

do {

cout << "Enter type of pyramid :\nGUIDE :\nNumber Pyramid - 0\nUpside-down Isosceles Pyramid - 1\nHollow Diamond - 2\nBottom-filled Diamond - 3\n\tChoice? ";

cin >> choice;

if (choice >= 4)

cout << "ERR : INPUT OUT OF BOUNDS(0-3)\n";

} while (choice >= 4);

switch (choice) {

case 0:

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

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

cout << ' ';

for (int j = 1; j < i + 2; ++j)

cout << j;

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

cout << j;

cout << '\n';

}

break;

case 1:

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

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

cout << ' ';

for (int j = 2 \* i - 1; j > 0; --j)

cout << '\*';

cout << '\n';

}

break;

case 2:

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

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

for (int i = lines/2; i >= 0; --i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

break;

case 3:

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

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

for (int i = lines/2; i >= 0; --i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

for (int j = 2\*i + 1; j > 0; --j)

cout << '\*';

cout << '\n';

}

break;

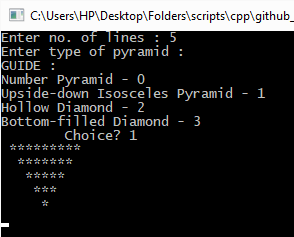
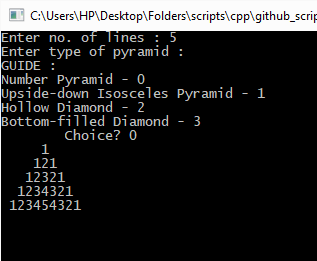
}

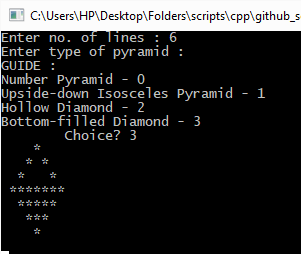
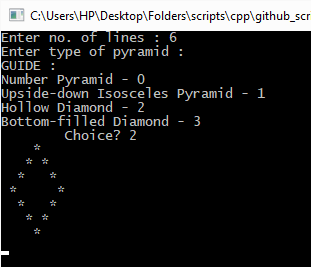
while (getchar() != '\n');

getchar();

}

Output :





1. Letter count

#include <iostream>

#include <stdio.h>

using namespace std;

bool isUpper (char ch) {

return ch >='A' and ch <= 'Z';

}

bool isLower (char ch) {

return ch >='a' and ch <= 'z';

}

int main() {

char str[1000];

cout << "Give your string :\n\t";

// cin << flush;

cin.getline(str, 1000, '\n');

int upper, lower, non\_alpha, words;

upper = lower = non\_alpha = words = 0;

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

if (isUpper(str[i]))

upper++;

else if (isLower(str[i]))

lower++;

else {

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

words++;

if (str[i + 1] == '.')

i++; non\_alpha++;

}

non\_alpha++;

}

}

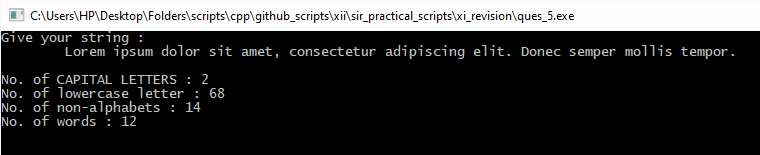
words++;

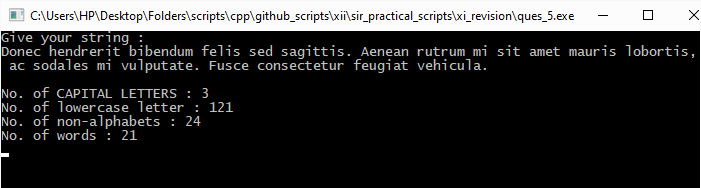
cout << "\nNo. of CAPITAL LETTERS : " << upper << "\nNo. of lowercase letter : " << lower << "\nNo. of non-alphabets : " << non\_alpha << "\nNo. of words : " << words << endl;

getchar();

}

Output :





1. cASE sWITCH

#include <iostream>

#include <stdio.h>

using namespace std;

bool isUpper (char ch) {

return ch >='A' and ch <= 'Z';

}

bool isLower (char ch) {

return ch >='a' and ch <= 'z';

}

char toUpper (char ch) {

if (isLower(ch))

return (ch - 0b00100000);

return ch;

}

char toLower (char ch) {

if (isUpper(ch))

return (ch + 0b00100000);

return ch;

}

int main() {

char str[1000];

cout << "Give your string :\n\t";

// cin << flush;

cin.getline(str, 1000, '\n');

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

if (isUpper(str[i]))

str[i] = toLower(str[i]);

else if (isLower(str[i]))

str[i] = toUpper(str[i]);

}

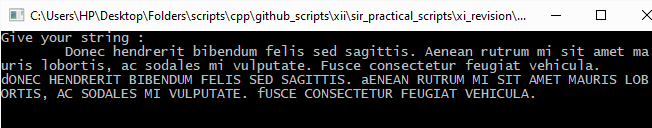
cout << str;

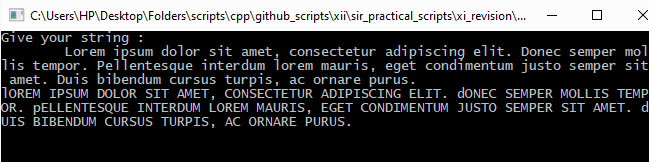
while (getchar() != '\n');

getchar();

}

Output :





1. Palindrome

#include <iostream>

#include <limits>

#include <string.h>

using namespace std;

bool isPalindrome(const char\* str) {

bool is\_palin = true;

unsigned len = strlen(str);

for (int i = len/2 - 1; i > -1 and is\_palin; --i)

is\_palin &= (str[i] == str[len - i - 1]);

return is\_palin;

}

int main() {

char \*str, \*concat\_str;

const char\* yes\_palin = "Hello Palindrome ";

const char\* no\_palin = "Hello Non-palindrome ";

unsigned len;

cout << "Enter string length : "; cin >> len;

str = new char[len + 1];

concat\_str = new char[len + 22];

cout << "Enter string : \n\t";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.clear();

cin.getline(str, len + 1, '\n');

if (isPalindrome(str))

strcopy(concat\_str, strcat(yes\_palin, str));

else

strcopy(concat\_str, strcat(no\_palin, str));

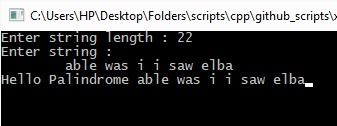
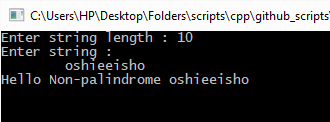
cout << concat\_str;

while (getchar() != '\n');

getchar();

}

Output :



1. Half-Half Array

#include <iostream>

using namespace std;

void swap(int arr[], int i\_1, int i\_2) {

int temp = arr[i\_1];

arr[i\_1] = arr[i\_2];

arr[i\_2] = temp;

}

int main() {

unsigned int arr\_size;

cout << "Array size? "; cin >> arr\_size;

int arr[arr\_size];

cout << "Array : "; for (int i = 0; i < arr\_size; ++i) cin >> arr[i];

for (int i = 0, j = arr\_size/2 + arr\_size % 2; j < arr\_size; i++, j++) {\

swap(arr, i , j);

}

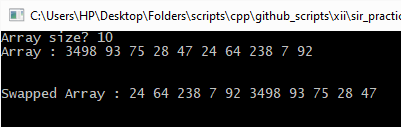
cout << "\n\nSwapped Array : "; for (int i = 0; i < arr\_size; ++i) cout << arr[i] << ' ';

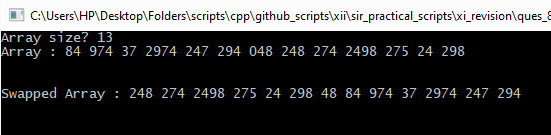
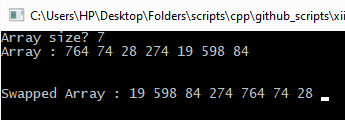
while (getchar() != '\n');

getchar();

}

Output :





1. De-duplication

#include <iostream>

using namespace std;

int main() {

int size\_arr, no\_of\_duplicates = 0;

cout << "Give size of array : "; cin >> size\_arr;

int arr[size\_arr];

bool arr\_tracker[size\_arr];

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

arr\_tracker[i] = true;

// Get input

cout << "Enter array : ";

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

cin >> arr[i];

cout << "\nOriginal Array : ";

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

cout << arr[i] << ' ';

// Mark and remove all duplicate entries

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

int in\_focus = arr[i];

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

if (arr[j] == in\_focus) {

arr\_tracker[j] = false;

no\_of\_duplicates++;

}

}

for (int i = 0, j; i < size\_arr - 1; ++i) {

if (arr\_tracker[i] == false) {

// find next non-duplicate entry

j = i + 1;

while (j < size\_arr) {

if(arr\_tracker[j] == true)

break;

else j++;

}

// No need to continue if is all duplcates till the end of the array

if (j == size\_arr) break;

// Swap values

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

// Swap duplication status

arr\_tracker[i] = true;

arr\_tracker[j] = false;

}

}

cout << "\nFinal Array : ";

for (int i = 0; i < size\_arr - no\_of\_duplicates; ++i)

cout << arr[i] << ' ';

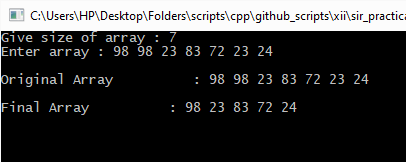
cout << '\n';

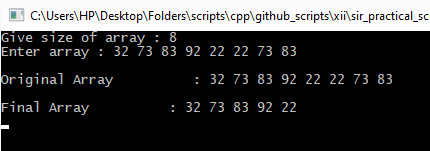
while(getchar() != '\n');

getchar();

}

Output :





1. Matrix-math

#include <iostream>

#include <iomanip>

#include <stdio.h>

using namespace std;

int main() {

int m, n;

cout << "Size of matrix (m x n)? "; cin >> m >> n;

int matrix[m][n];

cout << "Enter matrix : \n";

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

cout << '\t';

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

cin >> matrix[i][j];

}

cout << "\nColumn sums : \n\t";

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

int col\_sum = 0;

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

col\_sum += matrix[i][j];

cout << setw(10) << col\_sum << ' ';

}

cout << "\nRow sums : \n";

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

int row\_sum = 0;

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

row\_sum += matrix[i][j];

cout << '\t' << setw(10) << row\_sum << '\n';

}

cout << "\nMain diagonal sum : ";

int diag\_sum = 0;

for (int i = 0, j = 0; i < m and j < n; i++, j++)

diag\_sum += matrix[i][j];

cout << diag\_sum << '\n';

cout << "\nBack diagonal sum : ";

diag\_sum = 0;

for (int i = 0, j = n - 1; i < m and j >= 0; i++, j--)

diag\_sum += matrix[i][j];

cout << diag\_sum << '\n';

cout << "Transpose : \n";

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

cout << '\t';

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

cout << setw(10) << matrix[i][j];

cout << '\n';

}

cout << "\nUpper Triangle : \n";

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

cout << '\t';

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

if (i > j)

cout << setw(10) << '0';

else

cout << setw(10) << matrix[i][j];

}

cout << '\n';

}

cout << "\nLower Triangle : \n";

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

cout << '\t';

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

if (i < j)

cout << setw(10) << '0';

else

cout << setw(10) << matrix[i][j];

}

cout << '\n';

}

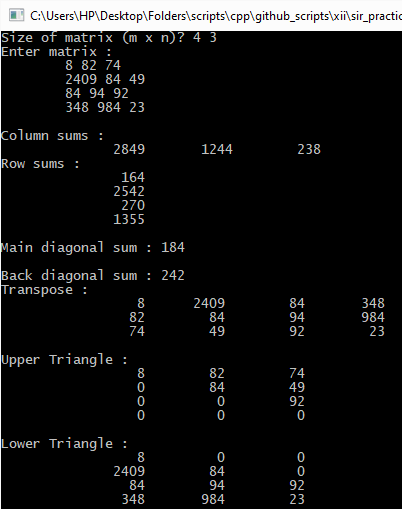
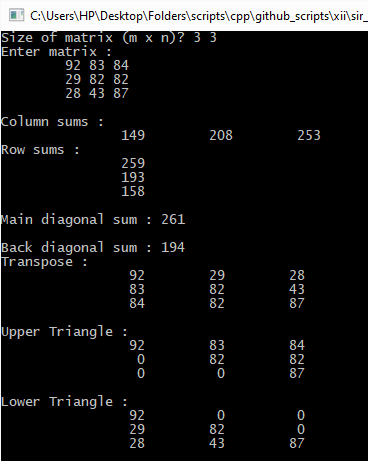
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. Temperature

#include <iostream>

using namespace std;

enum {MORNING, NOON, EVENING, NIGHT};

int main() {

float temp[3][7], min\_max\_temp[3][2] = {0, 0, 0, 0, 0, 0}, av\_temp[3] = {0,0,0}, av\_temp\_week;

cout << "Enter temperatures : \n";

for (int i = MORNING; i < NIGHT; ++i) {

cout << '\t';

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

cin >> temp[i][j];

cout << '\n';

}

for (int i = MORNING; i < NIGHT; ++i) {

min\_max\_temp[i][0] = temp[i][0];

min\_max\_temp[i][1] = temp[i][0];

}

for (int i = MORNING; i < NIGHT; ++i) {

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

if (min\_max\_temp[i][0] > temp[i][j])

min\_max\_temp[i][0] = temp[i][j];

if (min\_max\_temp[i][1] < temp[i][j])

min\_max\_temp[i][1] = temp[i][j];

av\_temp[i] += temp[i][j];

av\_temp\_week += temp[i][j];

}

av\_temp[i] /= 7;

}

av\_temp\_week /= 7 \* 3;

cout << "Minimum morning temperature : " << min\_max\_temp[MORNING][0] << '\n';

cout << "Minimum noon temperature : " << min\_max\_temp[NOON][0] << '\n';

cout << "Minimum evening temperature : " << min\_max\_temp[EVENING][0] << '\n';

cout << "Maximum morning temperature : " << min\_max\_temp[MORNING][1] << '\n';

cout << "Maximum noon temperature : " << min\_max\_temp[NOON][1] << '\n';

cout << "Maximum evening temperature : " << min\_max\_temp[EVENING][1] << '\n';

cout << "Average morning temperature : " << av\_temp[MORNING] << '\n';

cout << "Average noon temperature : " << av\_temp[NOON] << '\n';

cout << "Average evening temperature : " << av\_temp[EVENING] << '\n';

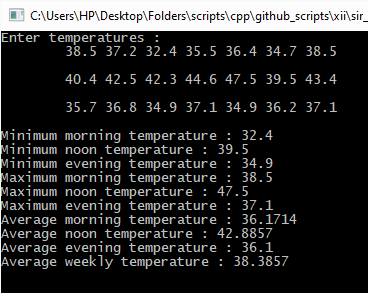
cout << "Average weekly temperature : " << av\_temp\_week << '\n';

while(getchar() != '\n');

getchar();

}

Output :



1. What the Square

#include <iostream>

#define SQR(x) x\*x

using namespace std;

int Sqr(int x) { return x \* x; }

inline int sqr(int x) { return x \* x;}

int main () {

unsigned int choice;

float n;

cout << "Welcome to Square-inator\n"

<< "========================\n";

cout << "Input number : ";

while(!(cin >> n)); // Check if cin goes into a fail state

cout << "Choose a squaring method :\n"

<< "\tMacro : 1\n"

<< "\tOutline function : 2\n"

<< "\tInline function : 3\n";

while(!(cin >> choice)); // Check if cin goes into a fail state

switch ( (choice - 1) % 3 + 1) {

case 1 :

cout << SQR(n);

break;

case 2 :

cout << Sqr(n);

break;

case 3 :

cout << sqr(n);

break;

}

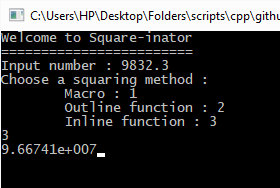
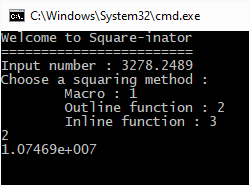
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. All Sort

#include <iostream>

#include <limits>

#include <string.h>

#include <time.h>

#include <stdlib.h>

using namespace std;

template <typename arr\_type>

void quick\_sort(arr\_type\* arr, int start, int end, bool descending = false) {

if (end - start <= 1 or start < 0) return;

int pivot\_point = end - 1; // default pivot location

int pivot\_location = start; // initial location of pivot

arr\_type temp; // for swapping

//partition array into two halves based on the pivot number

for (int i = start; i < end - 1; ++i) {

if (descending ^ (arr[i] < arr[pivot\_point]) ) {

// swap i and pivot\_location

temp = arr[pivot\_location];

arr[pivot\_location] = arr[i];

arr[i] = temp;

pivot\_location++;

}

}

// put pivot in it's place

temp = arr[pivot\_point];

arr[pivot\_point] = arr[pivot\_location];

arr[pivot\_location] = temp;

quick\_sort(arr, start, pivot\_location, descending);

quick\_sort(arr, pivot\_location + 1, end, descending);

return;

}

// template <> quick\_sort<char\*>

void quick\_sort(char\*\* arr, int start, int end, bool descending = false) {

if (end - start <= 1 or start < 0) return;

int pivot\_point = end - 1; // default pivot location

int pivot\_location = start; // initial location of pivot

char\* temp; // for swapping

//partition array into two halves based on the pivot number

for (int i = start; i < end - 1; ++i) {

if (descending ^ (strcmp(arr[i], arr[pivot\_point]) == -1) ) {

// swap i and pivot\_location

temp = arr[pivot\_location];

arr[pivot\_location] = arr[i];

arr[i] = temp;

pivot\_location++;

}

}

// put pivot in it's place

temp = arr[pivot\_point];

arr[pivot\_point] = arr[pivot\_location];

arr[pivot\_location] = temp; ;

quick\_sort(arr, start, pivot\_location, descending);

quick\_sort(arr, pivot\_location + 1, end, descending);

return;

}

int main () {

unsigned int no\_of\_elements, choice;

cout << "Give array datatype : \n"

<< "int : 1\n"

<< "float : 2\n"

<< "char : 3\n"

<< "string : 4\n";

cin >> choice;

cout << "Give no. of elements in array : "; cin >> no\_of\_elements;

choice = (choice - 1) % 4 + 1;

if (choice == 1) {

int\* arr = new int[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<int>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

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

} else if (choice == 2) {

float\* arr = new float[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<float>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

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

} else if (choice == 3) {

char\* arr = new char[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<char>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

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

} else if (choice == 4) {

char\*\* arr = new char\*[no\_of\_elements];

int max\_str\_size;

cout << "Maximum length of a string : "; cin >> max\_str\_size;

for (int i = 0; i < no\_of\_elements; ++i)

arr[i] = new char[max\_str\_size + 1];

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin.getline(arr[i], max\_str\_size + 1, '\n');

quick\_sort(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

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

}

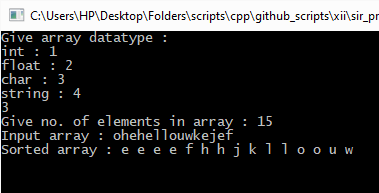
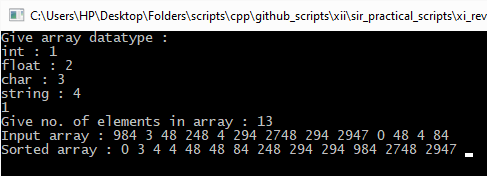
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. Full Power

#include <iostream>

#include <iomanip>

#include <limits>

using namespace std;

template <typename base\_type = double>

base\_type abs(base\_type x) {

if (x > 0)

return x;

else

return -x;

}

template <typename base\_type = double>

base\_type pow\_int(base\_type b, long long int e) {

if (e == 0 or b == 1) return 1;

base\_type num = 1, square\_base = b;

if (e < 0) square\_base = 1/b; // Takes care of inverse sign

long long int power = abs(e);

// Modular Exponentiation Algorithm

while (power > 0) {

if (power % 2)

num \*= square\_base;

square\_base \*= square\_base;

power /= 2;

}

return num;

}

template <typename base\_type = long long int>

base\_type fact(int x) {

base\_type res = 1;

for (int i = x; i > 1; --i)

res \*= i;

return res;

}

template <typename base\_type = double>

base\_type exp(base\_type x) {

base\_type a = 0, a\_old = 1;

for (int i = 0; a != a\_old; ++i) {

a\_old = a;

a += pow\_int<base\_type>(x, i) / fact<base\_type>(i);

}

return a;

}

template <typename base\_type = double>

base\_type ln(base\_type x) {

if (x <= 0) return numeric\_limits<base\_type>::quiet\_NaN();

base\_type a = 0, a\_old = 1;

for (int i = 1; a != a\_old; i+=2) {

a\_old = a;

a += pow\_int( ((x - 1)/(x + 1)), i) / i;

}

return 2\*a;

}

template <typename base\_type = double, typename exponent\_type = double>

base\_type pow(base\_type b, exponent\_type e) {

return exp(e \* ln(b));

}

template <typename base\_type = double, typename exponent\_type = double>

base\_type power(base\_type b, exponent\_type e = 1) {

return pow(b,e);

}

int main() {

long double x, n;

char pass\_exp;

cout << "Enter base and exponent : "; cin >> x >> n;

cout << "Pass exponent to function (y/n)? "; cin >> pass\_exp;

cout << "Result is ";

if (pass\_exp == 'n' or pass\_exp == 'N')

cout << setprecision(50) << std::fixed << power(x);

else

cout << setprecision(50) << std::fixed << power(x,n);

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

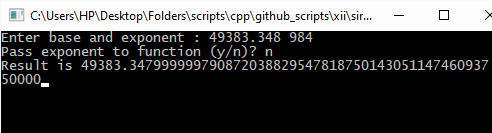
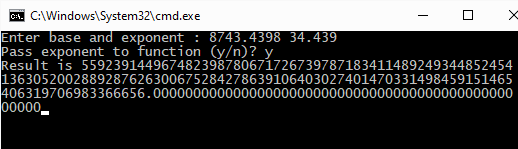
cin.clear();

getchar();

return 0;

}

Output :



1. Fibonacci

#include <iostream>

using namespace std;

float& min(float& a, float& b) {

if (a > b)

return b;

else return a;

}

int main() {

float k, l;

cout << "Enter values for k and l : "; cin >> k >> l;

cout << "\nk = " << k;

cout << "\nl = " << l;

min(k, l) \*= (1 + 0.1);

cout << "\nk = " << k;

cout << "\nl = " << l;

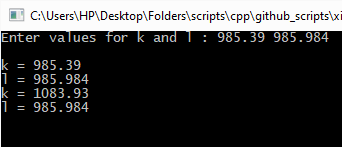
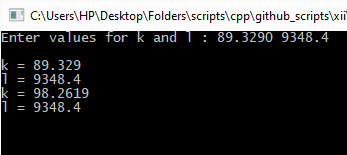
while(getchar() != '\n');

getchar();

return 0;

}

Output :



1. Assorted Students

#include <iostream>

#include <iomanip>

#include <limits>

#include <windows.h>

using namespace std;

// source - http://www.cplusplus.com/articles/4z18T05o/

#if defined \_WIN32

#include <windows.h>

void ClearScreen() {

HANDLE hStdOut;

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

DWORD count;

DWORD cellCount;

COORD homeCoords = { 0, 0 };

hStdOut = GetStdHandle( STD\_OUTPUT\_HANDLE );

if (hStdOut == INVALID\_HANDLE\_VALUE) return;

/\* Get the number of cells in the current buffer \*/

if (!GetConsoleScreenBufferInfo( hStdOut, &csbi )) return;

cellCount = csbi.dwSize.X \*csbi.dwSize.Y;

/\* Fill the entire buffer with spaces \*/

if (!FillConsoleOutputCharacter(

hStdOut,

(TCHAR) ' ',

cellCount,

homeCoords,

&count

)) return;

/\* Fill the entire buffer with the current colors and attributes \*/

if (!FillConsoleOutputAttribute(

hStdOut,

csbi.wAttributes,

cellCount,

homeCoords,

&count

)) return;

/\* Move the cursor home \*/

SetConsoleCursorPosition( hStdOut, homeCoords );

}

#elif defined (\_\_LINUX\_\_) || defined(\_\_gnu\_linux\_\_) || defined(\_\_linux\_\_) || defined (\_\_APPLE\_\_)

#include <unistd.h>

#include <term.h>

void ClearScreen() {

if (!cur\_term)

{

int result;

setupterm( NULL, STDOUT\_FILENO, &result );

if (result <= 0) return;

}

putp( tigetstr( "clear" ) );

}

#endif

void gotoxy(int x, int y) {

COORD c = { x, y };

SetConsoleCursorPosition( GetStdHandle(STD\_OUTPUT\_HANDLE) , c);

}

struct Student {

char name[20];

unsigned int marks;

};

void quickSort(Student\* arr, unsigned int start, unsigned int end, bool ascending = true) {

if (end <= start + 1 or arr == nullptr)

return;

int pivotIndex = start;

Student temp;

for (int i = start; i < end - 1; ++i) {

if (arr[i].marks < arr[end - 1].marks xor not ascending) {

temp = arr[i];

arr[i] = arr[pivotIndex];

arr[pivotIndex] = temp;

pivotIndex++;

}

}

temp = arr[pivotIndex];

arr[pivotIndex] = arr[end - 1];

arr[end - 1] = temp;

quickSort(arr, start, pivotIndex, ascending);

quickSort(arr, pivotIndex + 1, end, ascending);

return;

}

int main () {

Student stu\_arr[20];

int no\_of\_students;

bool ascending = false;

char choice;

do {

cout << "No. of students? ";

cin >> no\_of\_students;

} while (no\_of\_students > 20);

cout << "Enter student details : \n";

for (int i = 0; i < no\_of\_students; ++i) {

cout << "\tStudent " << i << " : \n";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin.clear();

cout << "\t\tName : "; cin.getline(stu\_arr[i].name, 20, '\n');

cout << "\t\tAggregate marks : "; cin >> stu\_arr[i].marks;

}

// Normal code

// cout << " S.No. Name Marks\n"

// << " ----- ------------------- -----\n";

// for (int i = 0; i < no\_of\_students; ++i) {

// cout << ' ' << setw(2) << i << ". ";

// cout << setw(19) << stu\_arr[i].name << " ";

// cout << setw(4) << stu\_arr[i].marks << '\n';

// }

cout << "\n\nDisplay in ascending (y/n)?";

cin >> choice;

if (choice == 'y' or choice == 'Y') ascending = true;

quickSort(stu\_arr, 0 , no\_of\_students, ascending);

ClearScreen();

gotoxy(1, 0);

cout << "S.No. Name Marks\n";

gotoxy(1, 1);

cout << "----- ------------------- -----\n";

for (int i = 0; i < no\_of\_students; ++i) {

gotoxy(1, 2 + i);

cout << setw(2) << (i + 1) << '.';

gotoxy(9, 2 + i);

cout << setw(19) << stu\_arr[i].name;

gotoxy(30, 2 + i);

cout << setw(4) << stu\_arr[i].marks;

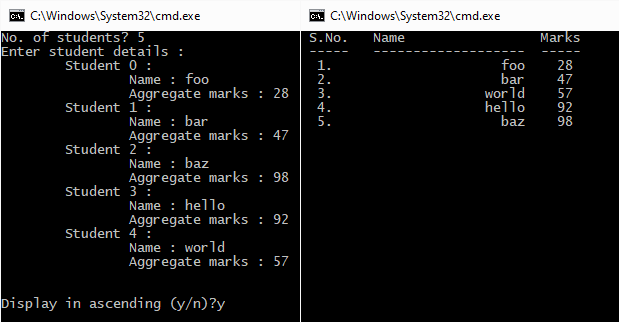
}

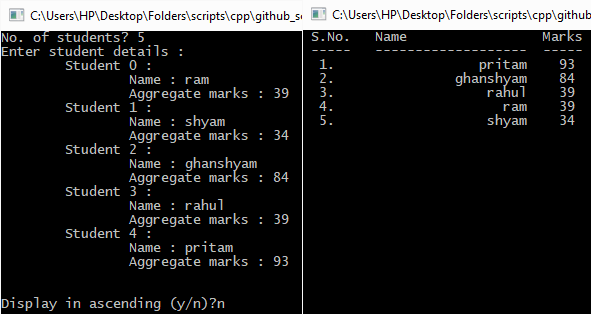
while (getchar() != '\n');

getchar();

}

Output :





1. Imperial Systems

#include <iostream>

using namespace std;

class FeetInches {

public :

int feet;

short inches;

FeetInches() {

this->feet = 0;

this->inches = 0;

}

FeetInches(int f, short i) {

this->feet = f;

this->inches = i;

this->Adjust();

}

FeetInches Adjust() {

if (this->inches > 0) {

this->feet += this->inches / 12;

this->inches %= 12;

} else {

this->feet += -1 + this->inches / 12;

this->inches = ((this->inches % 12) + 12) % 12;

}

}

FeetInches getInput(istream& input\_stream) {

while (!(input\_stream >> this->feet))

input\_stream.clear();

while (!(input\_stream >> this->inches))

input\_stream.clear();

this->Adjust();

}

FeetInches printSelf(ostream& display\_stream) {

display\_stream << this->feet << "\' " << this->inches << '\"';

}

FeetInches operator+(FeetInches f) {

return FeetInches(this->feet + f.feet, this->inches + f.inches);

}

FeetInches operator+=(FeetInches f) {

this->feet += f.feet;

this->inches += f.inches;

this->Adjust();

return \*this;

}

};

FeetInches add(FeetInches a, FeetInches b) {

return a + b;

}

int main () {

FeetInches dist1, dist2;

cout << "Enter dist1 (feet,inches) : "; dist1.getInput(cin);

cout << "Enter dist2 (feet,inches) : "; dist2.getInput(cin);

cout << "\n\ndist1 = "; dist1.printSelf(cout);

cout << "\ndist2 = "; dist2.printSelf(cout);

cout << "\ndist1 + dist2 = "; add(dist1, dist2).printSelf(cout);

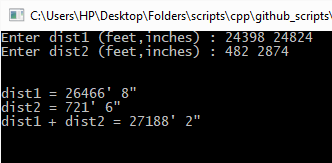
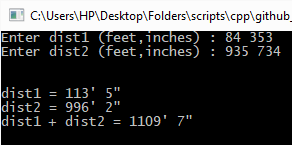
while(getchar() != '\n');

getchar();

return 0;

}

Output :



1. Streaming Students

#include <iostream>

#include <iomanip>

#include <limits>

using namespace std;

enum Stream {COMPUTER\_SCIENCE, ELECTRONICS, MECHANICAL, ELECTRICAL, CHEMICAL, CIVIL, NONE};

const char\* stream\_name(Stream stream) {

switch(stream) {

case COMPUTER\_SCIENCE :

return "Computer Science";

break;

case ELECTRONICS :

return "Electronics";

break;

case MECHANICAL :

return "Mechanical";

break;

case ELECTRICAL :

return "Electrical";

break;

case CHEMICAL :

return "Chemical";

break;

case CIVIL :

return "Civil";

break;

case NONE :

return "";

break;

}

}

class Student {

char name[30];

unsigned roll\_num;

unsigned marks[5];

Stream stream;

public :

Student() {

this->name[0] = '\0';

this->roll\_num = 0;

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

this->marks[i] = 0;

this->stream = NONE;

}

void getInput(istream& input\_stream, ostream& display\_stream, char\* pre\_str) {

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Name : ";

input\_stream.getline(this->name, 30, '\n');

display\_stream << pre\_str << "Roll no. : ";

while (!(input\_stream >> this->roll\_num)) {

input\_stream.ignore(1);

input\_stream.clear();

}

display\_stream << pre\_str << "Marks : ";

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

while (!(input\_stream >> this->marks[i])) {

input\_stream.ignore(1);

input\_stream.clear();

}

}

this->assignStream();

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num) {

display\_stream << setw(roll\_space\_num) << this->roll\_num << '.';

display\_stream << setw(name\_space\_num) << this->name;

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

display\_stream << setw(mark\_space\_num) << this->marks[i];

cout << setw(stream\_space\_num) << stream\_name(this->stream);

}

unsigned long total() {

unsigned long sum = 0;

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

sum += this->marks[i];

return sum;

}

float average() {

return this->total() / float(5);

}

Stream assignStream() {

float average\_marks = this->average();

if (average\_marks < 70)

this->stream = NONE;

else if (average\_marks < 75)

this->stream = CIVIL;

else if (average\_marks < 80)

this->stream = CHEMICAL;

else if (average\_marks < 85)

this->stream = ELECTRICAL;

else if (average\_marks < 90)

this->stream = MECHANICAL;

else if (average\_marks < 95)

this->stream = ELECTRONICS;

else

this->stream = COMPUTER\_SCIENCE;

return this->stream;

}

};

int main() {

Student stu\_arr[20];

unsigned short no\_of\_students;

cout << "No. of students? ";

do {

while (!(cin >> no\_of\_students)) cin.clear();

if (no\_of\_students > 20)

cout << "ERR : input out of range (0<, <=20)";

} while (no\_of\_students > 20);

cout << "Enter student details : ";

for (int i = 0; i < no\_of\_students; ++i) {

char format\_string[] = "\t\t";

cout << "\n\tStudent " << (i + 1) << " : \n";

stu\_arr[i].getInput(cin, cout, format\_string);

}

cout << "Student Data : \n";

cout << " Sr.no. Name Phys Chem Math Engl Comp Stream \n"

<< " ------ ----------------------------- ---- ---- ---- ---- ---- ----------------\n";

for (int i = 0; i < no\_of\_students; ++i) {

stu\_arr[i].printSelf(cout, 31, 6, 6, 18);

cout << '\n';

}

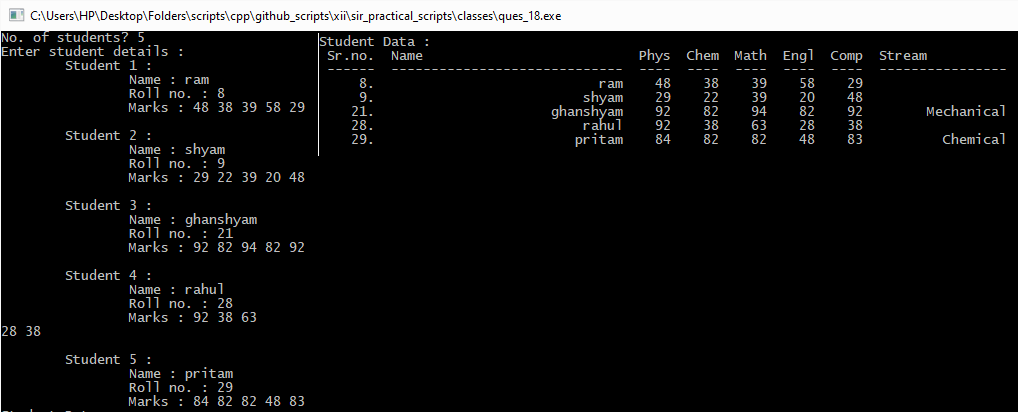
while (getchar() != '\n');

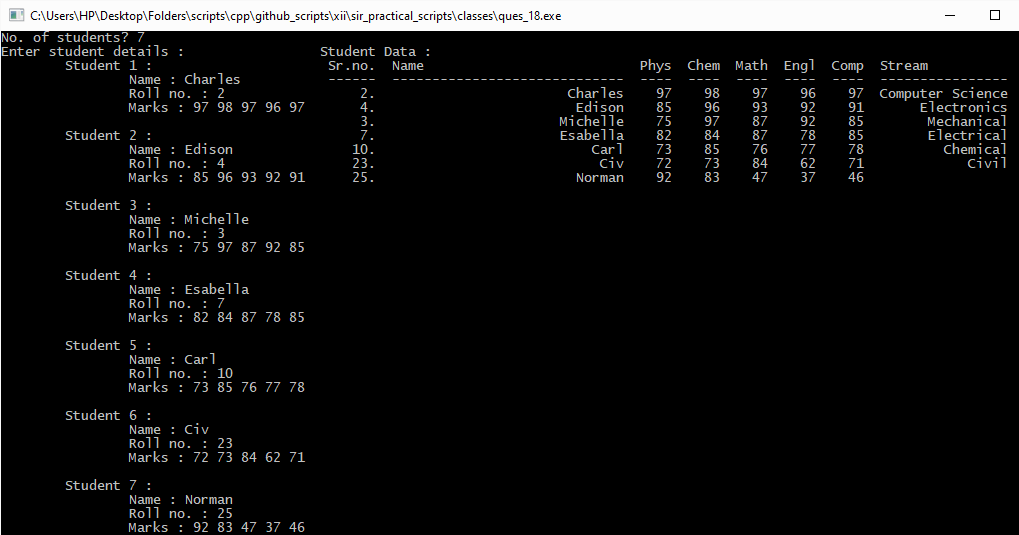
getchar();

return 0;

}

Output :





1. Divided they stand

#include <iostream>

using namespace std;

long long HCF(long long a, long long b) {

long long temp;

if (a < b) {

temp = a;

a = b;

b = temp;

}

while (a % b != 0ll) {

temp = b;

b = a % b;

a = temp;

}

return b;

}

inline long long LCM(long long a, long long b) {

return a \* (b / HCF(a,b));

}

class Fraction {

public :

long long numerator, denominator;

Fraction() {

this->numerator = 0;

this->denominator = 1;

}

Fraction(long long num, long long den) {

long long hcf = HCF(num,den);

this->numerator = num / hcf;

if (den == 0)

this->denominator = 1;

else

this->denominator = den / hcf;

if (this->denominator < 0) {

this->denominator \*= -1;

this->numerator \*= -1;

}

}

Fraction(long long k) {

\*this = Fraction(k,1);

// this->Fraction(k,1);

}

Fraction(long double d) {

\*this = Fraction(d \* 1e8, 1e8);

}

Fraction add(const Fraction& a) const {

long long den\_hcf = HCF(this->denominator, a.denominator);

return Fraction(this->numerator \* (a.denominator / den\_hcf) + a.numerator \* (this->denominator / den\_hcf),

LCM(this->denominator, a.denominator));

}

Fraction subtract(const Fraction& s) const {

long long den\_hcf = HCF(this->denominator, s.denominator);

return Fraction(this->numerator \* (s.denominator / den\_hcf) - s.numerator \* (this->denominator \* den\_hcf),

LCM(this->denominator, s.denominator));

}

Fraction multiply(const Fraction& m) const {

return Fraction(this->numerator \* m.numerator, this->denominator \* m.denominator);

}

Fraction divide(const Fraction& d) const {

return Fraction(this->numerator \* d.denominator, this->denominator \* d.numerator);

}

operator long double() {

return (long double)(this->numerator) / this->denominator;

}

inline friend Fraction operator+(const Fraction& a, const Fraction& b) {

return a.add(b);

}

inline friend Fraction operator-(const Fraction& a, const Fraction& b) {

return a.subtract(b);

}

inline friend Fraction operator\*(const Fraction& a, const Fraction& b) {

return a.multiply(b);

}

inline friend Fraction operator/(const Fraction& a, const Fraction& b) {

return a.divide(b);

}

inline friend Fraction operator+=(Fraction& a, const Fraction& b) {

return a = a.add(b);

}

inline friend Fraction operator-=(Fraction& a, const Fraction& b) {

return a = a.subtract(b);

}

inline friend Fraction operator\*=(Fraction& a, const Fraction& b) {

return a = a.multiply(b);

}

inline friend Fraction operator/=(Fraction& a, const Fraction& b) {

return a = a.divide(b);

}

friend istream& operator>>(istream& input\_stream, Fraction& f) {

while (!(input\_stream >> f.numerator)) {

input\_stream.ignore(1);

input\_stream.clear();

}

while (!(input\_stream >> f.denominator)) {

input\_stream.ignore(1);

input\_stream.clear();

}

f = Fraction(f.numerator, f.denominator);

return input\_stream;

}

friend ostream& operator<<(ostream& display\_stream, const Fraction f) {

display\_stream << f.numerator << '/' << f.denominator;

return display\_stream;

}

};

int main() {

Fraction r1, r2;

cout << "Enter r1 : "; cin >> r1;

cout << "Enter r2 : "; cin >> r2;

cout << "\nr1 = " << r1;

cout << "\nr2 = " << r2;

cout << "\n\nr1 + r2 = " << r1 + r2;

cout << "\nr1 - r2 = " << r1 - r2;

cout << "\nr1 \* r2 = " << r1 \* r2;

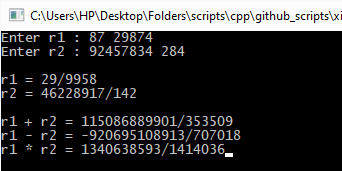
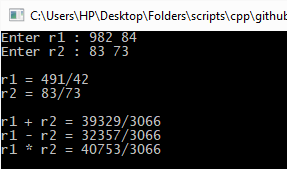
while (getchar() != '\n');

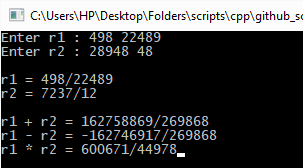
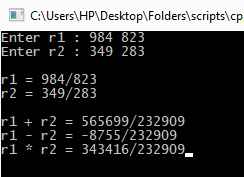
getchar();

return 0;

}

Output :





1. To the Point

#include <iostream>

#include <math.h>

using namespace std;

class Point {

public :

float x, y;

Point (float x, float y) {

this->x = x;

this->y = y;

}

Point() {

\*this = Point(0, 0);

}

Point(const Point& p) {

\*this = Point(p.x, p.y);

}

float sqrDistanceFrom(const Point& b) const {

return ((this->x - b.x)\*(this->x - b.x) + (this->y - b.y)\*(this->y - b.y));

}

float distanceFrom(const Point& b) const {

return sqrt( this->sqrDistanceFrom(b) );

}

float dotProduct(const Point& b) const {

return (this->x \* b.x + this->y \* b.y);

}

float crossProduct(const Point& b) const {

return (this->x \* b.y - this->y \* b.x);

}

friend istream& operator>>(istream& input\_stream, Point& p) {

while (!(input\_stream >> p.x)) {

input\_stream.ignore(1);

input\_stream.clear();

}

while (!(input\_stream >> p.y)) {

input\_stream.ignore(1);

input\_stream.clear();

}

return input\_stream;

}

friend ostream& operator<<(ostream& display\_stream, const Point& p) {

display\_stream << '(' << p.x << ", " << p.y << ')';

return display\_stream;

}

};

class Triangle {

public :

Point points[3];

Triangle(float x1, float y1, float x2, float y2, float x3, float y3) {

this->points[0] = Point(x1, y1);

this->points[1] = Point(x2, y2);

this->points[2] = Point(x3, y3);

}

Triangle(const Point& p1, const Point& p2, const Point& p3) {

this->points[0] = p1;

this->points[1] = p2;

this->points[2] = p3;

}

Triangle() {

\*this = Triangle(0, 0, 0, 0, 0, 0);

}

float perimeter() const {

return this->points[0].distanceFrom(this->points[1])

+ this->points[1].distanceFrom(this->points[2])

+ this->points[2].distanceFrom(this->points[0]);

}

float area() const {

return 0.5 \* abs(this->points[0].crossProduct(this->points[1])

+ this->points[1].crossProduct(this->points[2])

+ this->points[2].crossProduct(this->points[0]));

}

inline friend istream& operator>>(istream& input\_stream, Triangle& t) {

return input\_stream >> t.points[0] >> t.points[1] >> t.points[2];

}

inline friend ostream& operator<<(ostream& display\_stream, const Triangle& t) {

return display\_stream << "( " << t.points[0] << ", " << t.points[1] << ", " << t.points[2] << ")";

}

};

int main() {

Triangle quadrilateral;

cout << "Enter the points of the triangle : "; cin >> quadrilateral;

cout << "\nThe points you have entered are " << quadrilateral;

cout << "\n\nPerimeter = " << quadrilateral.perimeter();

cout << "\nArea = " << quadrilateral.area();

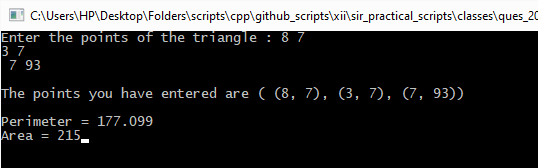
while(getchar() != '\n');

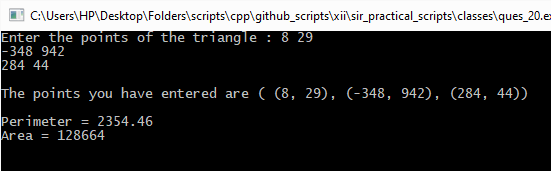
getchar();

return 0;

}

Output :





1. Part-Time Students

#include <iostream>

#include <iomanip>

using namespace std;

enum Stream {COMPUTER\_SCIENCE, ELECTRONICS, MECHANICAL, ELECTRICAL, CHEMICAL, CIVIL, NONE};

const char\* stream\_name(Stream stream) {

switch(stream) {

case COMPUTER\_SCIENCE :

return "Computer Science";

break;

case ELECTRONICS :

return "Electronics";

break;

case MECHANICAL :

return "Mechanical";

break;

case ELECTRICAL :

return "Electrical";

break;

case CHEMICAL :

return "Chemical";

break;

case CIVIL :

return "Civil";

break;

case NONE :

return "";

break;

}

}

class Student {

char name[30];

unsigned roll\_num;

unsigned marks[5];

Stream stream;

public :

Student() {

this->name[0] = '\0';

this->roll\_num = 0;

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

this->marks[i] = 0;

this->stream = NONE;

}

void getInput(istream& input\_stream, ostream& display\_stream, const char\* pre\_str) {

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Name : ";

input\_stream.getline(this->name, 30, '\n');

display\_stream << pre\_str << "Roll no. : ";

while (!(input\_stream >> this->roll\_num)) {

input\_stream.ignore(1);

input\_stream.clear();

}

display\_stream << pre\_str << "Marks : ";

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

while (!(input\_stream >> this->marks[i])) {

input\_stream.ignore(1);

input\_stream.clear();

}

}

this->assignStream();

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num) {

display\_stream << setw(roll\_space\_num) << this->roll\_num << '.';

display\_stream << setw(name\_space\_num) << this->name;

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

display\_stream << setw(mark\_space\_num) << this->marks[i];

cout << setw(stream\_space\_num) << stream\_name(this->stream);

}

unsigned long total() {

unsigned long sum = 0;

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

sum += this->marks[i];

return sum;

}

float average() {

return this->total() / float(5);

}

Stream assignStream() {

float average\_marks = this->average();

if (average\_marks < 70)

this->stream = NONE;

else if (average\_marks < 75)

this->stream = CIVIL;

else if (average\_marks < 80)

this->stream = CHEMICAL;

else if (average\_marks < 85)

this->stream = ELECTRICAL;

else if (average\_marks < 90)

this->stream = MECHANICAL;

else if (average\_marks < 95)

this->stream = ELECTRONICS;

else

this->stream = COMPUTER\_SCIENCE;

return this->stream;

}

};

class WorkingStudent : public Student {

char job[20];

char office\_address[50];

public :

WorkingStudent() : Student() {

this->job[0] = '\0';

this->office\_address[0] = '\0';

}

void getInput(istream& input\_stream, ostream& display\_stream, const char\* pre\_str) {

this->Student::getInput(input\_stream, display\_stream, pre\_str);

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Job : ";

input\_stream.getline(this->job, 20, '\n');

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Office Address : ";

input\_stream.getline(this->office\_address, 50, '\n');

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num, int job\_space\_num, int off\_addr\_space\_num) {

this->Student::printSelf(display\_stream, name\_space\_num, roll\_space\_num, mark\_space\_num, stream\_space\_num);

display\_stream << setw(job\_space\_num) << this->job;

display\_stream << setw(off\_addr\_space\_num) << this->office\_address;

}

};

int main() {

WorkingStudent stu\_arr[20];

unsigned short no\_of\_students;

cout << "No. of students? ";

do {

while (!(cin >> no\_of\_students)) cin.clear();

if (no\_of\_students > 20)

cout << "ERR : input out of range (0<, <=20)";

} while (no\_of\_students > 20);

cout << "Enter student details : ";

for (int i = 0; i < no\_of\_students; ++i) {

char format\_string[] = "\t\t";

cout << "\n\tStudent " << (i + 1) << " : \n";

stu\_arr[i].getInput(cin, cout, format\_string);

}

cout << "Student Data : \n";

cout << " Sr.no. Name Phys Chem Math Engl Comp Stream Job Office Address\n"

<< " ------ ----------------------------- ---- ---- ---- ---- ---- ---------------- -------------------- --------------------------------------------------\n";

for (int i = 0; i < no\_of\_students; ++i) {

stu\_arr[i].printSelf(cout, 31, 6, 6, 18, 22, 52);

cout << '\n';

}

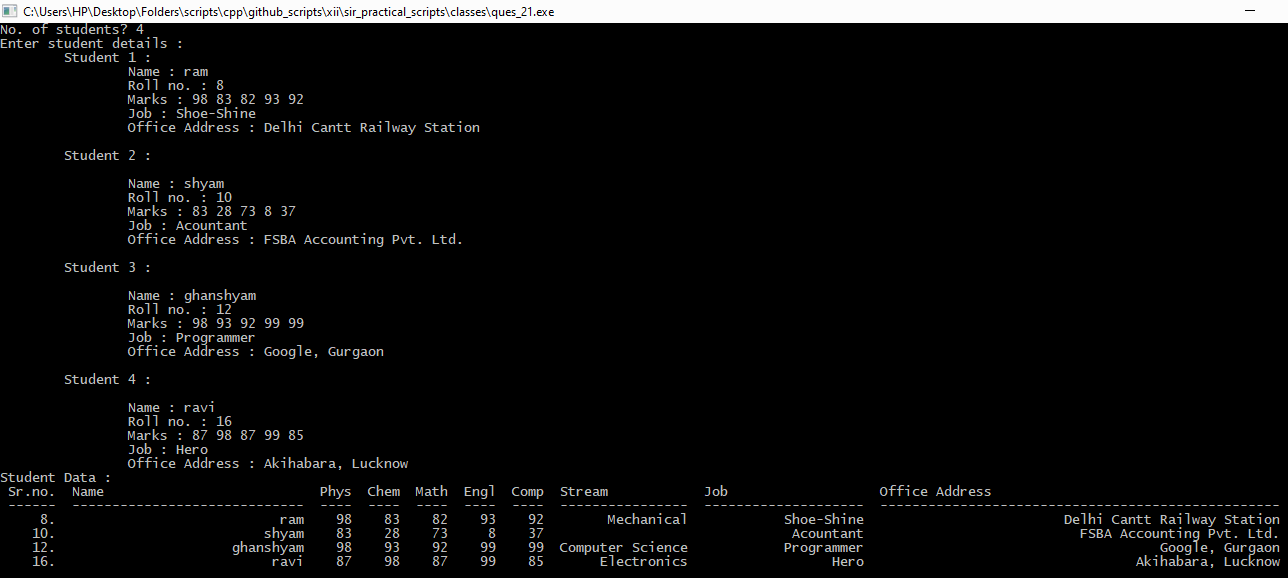
while(getchar() != '\n');

getchar();

return 0;

}

Output :



For text file-handling :

* lorem\_ipsum.txt :

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec semper mollis tempor. Pellentesque interdum lorem mauris, eget condimentum justo semper sit amet. Duis bibendum cursus turpis, ac ornare purus. Donec mollis neque eget faucibus consectetur. Proin iaculis sit amet massa et molestie. Sed consectetur est a scelerisque varius. Quisque gravida metus tortor, sed pulvinar ipsum cursus quis. Nullam rhoncus molestie diam, quis tempor nibh tristique at. Aliquam placerat bibendum lacus, non ornare nulla ornare id. Integer sit amet eros nulla. Sed et posuere libero, eget faucibus justo.

Integer faucibus ligula vitae diam congue, nec porttitor turpis porta. Phasellus non tempus arcu. Nulla diam dui, interdum at massa vel, tempus rutrum tortor. Fusce a tempor tortor. Maecenas non auctor massa, fermentum fermentum nisi. Etiam facilisis arcu vitae consectetur sagittis. Donec eget facilisis purus, eget eleifend nulla. Quisque venenatis, justo quis convallis facilisis, lacus lorem facilisis nunc, vitae venenatis velit tortor id est. Nam dictum eros ac risus facilisis pulvinar lobortis nec massa. Phasellus a pulvinar dolor, id iaculis ligula. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Ut non tortor hendrerit, finibus neque quis, euismod urna. Vivamus eleifend lorem pellentesque gravida vestibulum. Etiam ac quam tincidunt, scelerisque arcu quis, viverra massa. Vivamus sollicitudin maximus nulla, nec condimentum sapien efficitur sit amet.

Integer fermentum augue quis nisl maximus rutrum. Nunc nec pharetra leo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Quisque finibus porta est, non porttitor justo cursus ut. Suspendisse eleifend tempus massa, eget venenatis odio finibus ut. Donec ante orci, ullamcorper eget volutpat ut, auctor ornare massa. Fusce ornare sem nulla, id vulputate enim consectetur quis. Donec turpis augue, dignissim in nisi vitae, blandit dignissim tortor. Proin nec elementum ipsum, nec efficitur mauris.

Donec hendrerit bibendum felis sed sagittis. Aenean rutrum mi sit amet mauris lobortis, ac sodales mi vulputate. Fusce consectetur feugiat vehicula. Vivamus at dolor nec mauris scelerisque imperdiet. In accumsan, erat eu dapibus viverra, lacus nibh sodales purus, id suscipit risus lorem quis leo. Pellentesque et quam turpis. Aenean libero justo, pharetra ac neque convallis, efficitur sollicitudin ex. Cras auctor posuere sem sed hendrerit. Nullam eu metus et neque fringilla molestie at condimentum ex. Phasellus vitae leo placerat velit aliquet tincidunt. Integer volutpat ut ante nec porttitor.

Nunc nulla dolor, accumsan nec maximus id, gravida ut risus. Maecenas ullamcorper a justo quis venenatis. Aenean cursus libero et dolor dignissim imperdiet. Suspendisse feugiat dui sed rutrum semper. Mauris interdum non orci ut pulvinar. Quisque porta urna eget lacus tempor rutrum. Aenean velit nunc, rhoncus ac venenatis a, porta ut ligula. Etiam ut neque sed neque pellentesque volutpat eu ut magna. Vestibulum a leo eu orci imperdiet consequat. Ut ac est at tortor porta tincidunt ac sit amet libero. Sed varius congue arcu vel luctus. Vestibulum quis dolor vitae mauris facilisis feugiat. In vehicula dolor quis tristique molestie.

1. Text Census

#include <iostream>

#include <fstream>

using namespace std;

bool isUpperCase(char c) {

return (c <= 'Z' and c >= 'A');

}

bool isLowerCase(char c) {

return (c <= 'z' and c >= 'a');

}

bool isAlpha(char c) {

return isUpperCase(c) or isLowerCase(c);

}

bool isDigit(char c) {

return (c <= '9' and c >= '0');

}

bool isAlphaNum(char c) {

return isAlpha(c) or isDigit(c);

}

bool isVowel(char c) {

switch (c) {

case 'a' : case 'e' : case 'i' : case 'o' : case 'u' : case 'A' : case 'E' : case 'I' : case 'O' : case 'U' :

return true;

break;

default :

return false;

break;

}

}

int main() {

char file\_name[51];

ifstream input\_file;

cout << "Enter file name (max 50 char) : "; cin.getline(file\_name, 51, '\n');

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cout << "ERR : File does not exist\n";

return 0;

}

input\_file.seekg(0, ios::beg);

long long file\_length = -input\_file.tellg();

input\_file.seekg(0, ios::end);

file\_length += input\_file.tellg();

input\_file.seekg(0, ios::beg);

char ch, prev\_ch;

int no\_of\_blanks = 0, no\_of\_lines = 0, no\_of\_up = 0, no\_of\_low = 0, no\_of\_lines\_start\_up = 0, no\_of\_words = 0, no\_of\_digits = 0, no\_of\_words\_end\_vow = 0;

bool word\_started = false;

//For first character

input\_file.get(ch);

input\_file.seekg(1, ios::beg);

if (isUpperCase(ch)) {

no\_of\_up++; word\_started = true;

no\_of\_lines\_start\_up++;

} else if (isLowerCase(ch)) {

no\_of\_low++; word\_started = true;

} else if (isDigit(ch)) {

no\_of\_digits++;

} else if (ch == '\n' or ch == '\r') {

no\_of\_lines++;

no\_of\_blanks++;

}

while (not input\_file.eof()) {

prev\_ch = ch;

input\_file.read(&ch, 1);

if (isUpperCase(ch)) {

no\_of\_up++; word\_started = true;

if (prev\_ch == '\n' or prev\_ch == '\r')

no\_of\_lines\_start\_up++;

} else if (isLowerCase(ch)) {

no\_of\_low++; word\_started = true;

} else if (isDigit(ch)) {

no\_of\_digits++;

} else if (ch == '\n' or ch == '\r') {

no\_of\_lines++;

if (word\_started)

no\_of\_words++;

word\_started = false;

if (isVowel( prev\_ch ))

no\_of\_words\_end\_vow++;

else if (prev\_ch == '\n' or prev\_ch == '\r')

no\_of\_blanks++;

} else if (ch == ' ') {

if (word\_started)

no\_of\_words++;

word\_started = false;

if (isVowel( prev\_ch ))

no\_of\_words\_end\_vow++;

} else if (not isAlphaNum(ch)) {

if (word\_started)

no\_of\_words++;

word\_started = false;

}

}

input\_file.clear();

no\_of\_lines++;

if (word\_started)

no\_of\_words++;

word\_started = false;

if(isVowel(input\_file.get())) {

no\_of\_words\_end\_vow++;

}

cout << "\n\nNo . of blanks = " << no\_of\_blanks;

cout << "\nNo . of lines = " << no\_of\_lines;

cout << "\nNo . of UPPERCASE letters = " << no\_of\_up;

cout << "\nNo . of lowercase letters = " << no\_of\_low;

cout << "\nNo . of lines starting with a Capital letter = " << no\_of\_lines\_start\_up;

cout << "\nNo . of words = " << no\_of\_words;

cout << "\nNo . of digits = " << no\_of\_digits;

cout << "\nNo . of words ending with a vowel = " << no\_of\_words\_end\_vow;

input\_file.close();

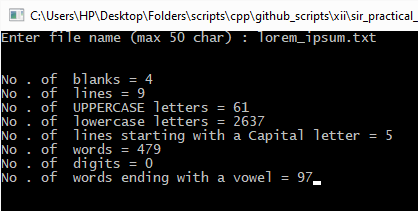
while (cin.get() != '\n');

cin.get();

return 0;

}

Output :



1. Substring finder

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream input\_file;

char file\_name[100], substring[100];

bool present = false;

cout << "Enter filename for input : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cerr << "ValueError : No such file as '" << file\_name << "'\n";

return 1;

}

input\_file.seekg(0, ios::beg);

cout << "Enter string to search : "; cin.getline(substring, 100);

while (not input\_file.eof() and not present) {

char first\_char;

input\_file.read(&first\_char, 1);

if (first\_char == substring[0]) {

present = true;

for (int i = 1; substring[i] != '\0' and present; ++i) {

input\_file.read(&first\_char, 1);

present &= substring[i] == first\_char;

}

}

}

if (present)

cout << "Given substring '" << substring << "' is present in '" << file\_name << "'\n";

else

cout << "Given substring '" << substring << "' is not present in '" << file\_name << "'\n";

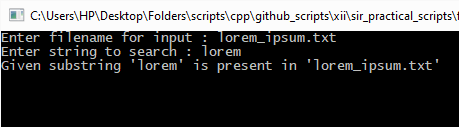
while (cin.get() != '\n');

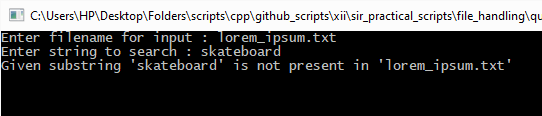
cin.get();

return 0;

}

Output :





1. Vowel terminator

#include <iostream>

#include <fstream>

using namespace std;

bool isVowel(char c) {

switch (c) {

case 'a' : case 'e' : case 'i' : case 'o' : case 'u' : case 'A' : case 'E' : case 'I' : case 'O' : case 'U' :

return true;

break;

default :

return false;

break;

}

}

int main() {

ifstream input\_file;

ofstream output\_file("copy.txt", ios::out);

char file\_name[100], word[100];

cout << "Enter filename for input : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cerr << "ValueError : No such file as '" << file\_name << "'\n";

return 1;

}

input\_file.seekg(0, ios::beg);

while (not input\_file.eof() ) {

input\_file >> word;

int i;

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

if (not isVowel(word[i - 1]))

output\_file << word << ' ';

}

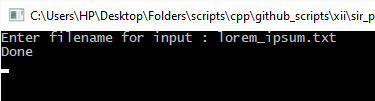
cout << "Done\n";

while (cin.get() != '\n');

cin.get();

}

Output :



* copy.txt :

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec semper mollis tempor. interdum lorem mauris, eget condimentum semper sit amet. Duis bibendum cursus turpis, ac purus. Donec mollis eget faucibus consectetur. Proin iaculis sit amet et molestie. Sed consectetur est varius. metus tortor, sed pulvinar ipsum cursus quis. Nullam rhoncus diam, quis tempor nibh at. Aliquam placerat bibendum lacus, non id. Integer sit amet eros nulla. Sed et libero, eget faucibus justo. Integer faucibus diam congue, nec porttitor turpis porta. Phasellus non tempus arcu. diam dui, interdum at vel, tempus rutrum tortor. tempor tortor. Maecenas non auctor massa, fermentum fermentum nisi. Etiam facilisis consectetur sagittis. Donec eget facilisis purus, eget eleifend nulla. venenatis, quis convallis facilisis, lacus lorem facilisis nunc, venenatis velit tortor id est. Nam dictum eros ac risus facilisis pulvinar lobortis nec massa. Phasellus pulvinar dolor, id iaculis ligula. habitant senectus et netus et fames ac turpis egestas. Ut non tortor hendrerit, finibus quis, euismod urna. Vivamus eleifend lorem vestibulum. Etiam ac quam tincidunt, quis, massa. Vivamus sollicitudin maximus nulla, nec condimentum sapien efficitur sit amet. Integer fermentum quis nisl maximus rutrum. Nunc nec leo. Vestibulum ipsum primis in faucibus luctus et ultrices Curae; finibus est, non porttitor cursus ut. eleifend tempus massa, eget venenatis finibus ut. Donec orci, ullamcorper eget volutpat ut, auctor massa. sem nulla, id enim consectetur quis. Donec turpis augue, dignissim in vitae, blandit dignissim tortor. Proin nec elementum ipsum, nec efficitur mauris. Donec hendrerit bibendum felis sed sagittis. Aenean rutrum sit amet mauris lobortis, ac sodales vulputate. consectetur feugiat vehicula. Vivamus at dolor nec mauris imperdiet. In accumsan, erat dapibus viverra, lacus nibh sodales purus, id suscipit risus lorem quis leo. et quam turpis. Aenean justo, ac convallis, efficitur sollicitudin ex. Cras auctor sem sed hendrerit. Nullam metus et at condimentum ex. Phasellus placerat velit aliquet tincidunt. Integer volutpat ut nec porttitor. Nunc dolor, accumsan nec maximus id, ut risus. Maecenas ullamcorper quis venenatis. Aenean cursus et dolor dignissim imperdiet. feugiat sed rutrum semper. Mauris interdum non ut pulvinar. eget lacus tempor rutrum. Aenean velit nunc, rhoncus ac venenatis a, ut ligula. Etiam ut sed volutpat ut magna. Vestibulum imperdiet consequat. Ut ac est at tortor tincidunt ac sit amet libero. Sed varius vel luctus. Vestibulum quis dolor mauris facilisis feugiat. In dolor quis molestie.

1. Fibonacci

#include <iostream>

#include <fstream>

using namespace std;

int main() {

fstream input\_file, temp\_file;

char file\_name[100];

char old\_ch, new\_ch;

cout << "Enter input filename : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

temp\_file.open(".temp.txt", ios::out);

if (not input\_file.is\_open()) {

cerr << "ValueError : Invalid filename given";

return 1;

}

input\_file.seekg(0, ios::beg);

temp\_file.seekp(0, ios::beg);

cout << "Enter the two characters to find and replace : ";

cin >> old\_ch >> new\_ch;

while (not input\_file.eof()) {

char ch;

input\_file.read(&ch, 1);

if (ch == old\_ch)

temp\_file.write(&new\_ch, 1);

else

temp\_file.write(&ch, 1);

}

input\_file.close(); input\_file.open(file\_name, ios::out);

temp\_file.close(); temp\_file.open(".temp.txt", ios::in);

input\_file.seekp(0, ios::beg);

temp\_file.seekg(0, ios::beg);

while (not temp\_file.eof()) {

char ch;

temp\_file.read(&ch, 1);

input\_file.write(&ch, 1);

}

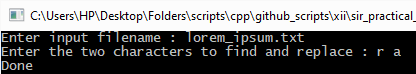
cout << "Done";

while( cin.get() != '\n');

cin.get();

}

Output :



1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

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

cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :