**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

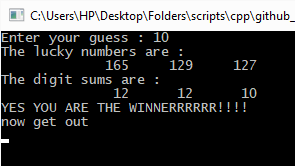
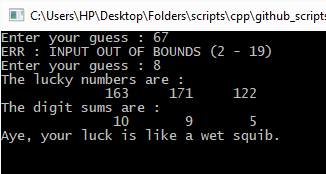
cout << "\nAye, your luck is like a wet squib.";

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)

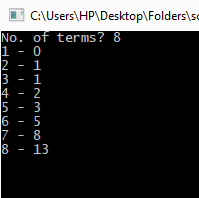
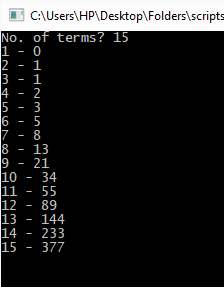
cout << i << " - " << fibonacci(i) << endl;

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;

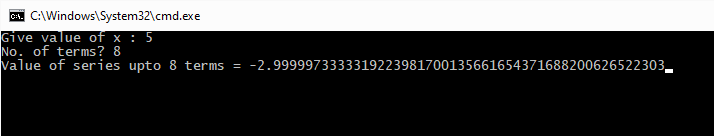
cout << "Value of series upto " << n << " terms = " << setprecision(50) << sum;

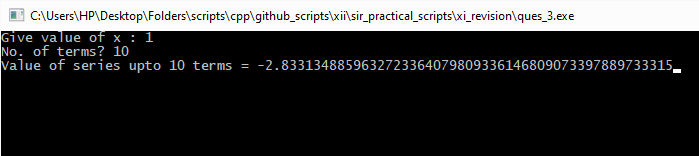
while (getchar() != '\n');

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;

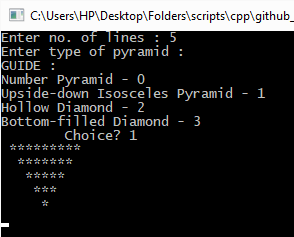
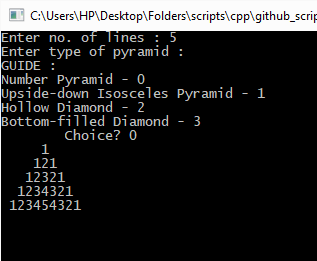
}

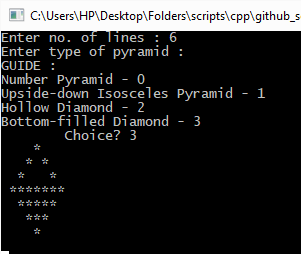
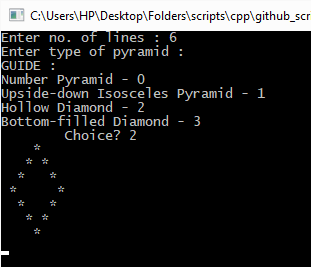
while (getchar() != '\n');

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++;

}

}

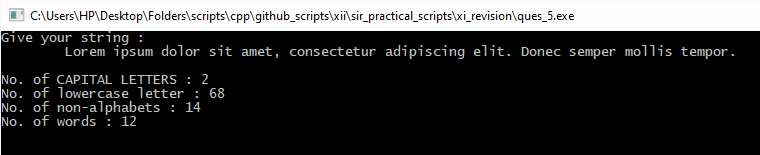
words++;

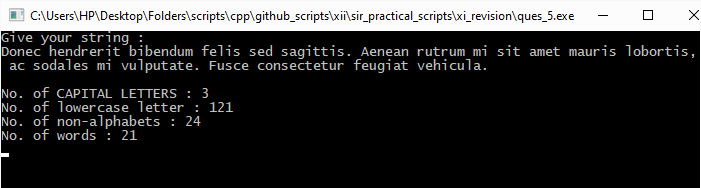
cout << "\nNo. of CAPITAL LETTERS : " << upper << "\nNo. of lowercase letter : " << lower << "\nNo. of non-alphabets : " << non\_alpha << "\nNo. of words : " << words << endl;

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]);

}

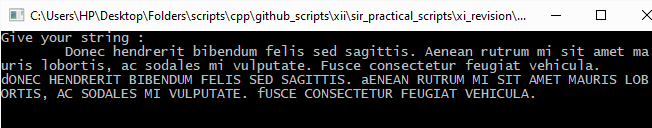
cout << str;

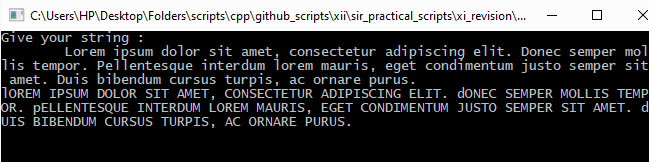
while (getchar() != '\n');

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));

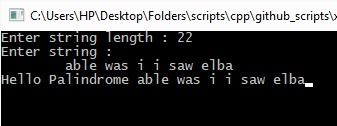
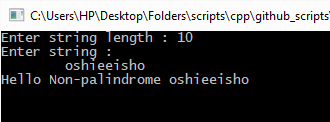
cout << concat\_str;

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);

}

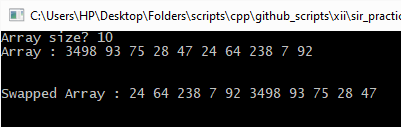
cout << "\n\nSwapped Array : "; for (int i = 0; i < arr\_size; ++i) cout << arr[i] << ' ';

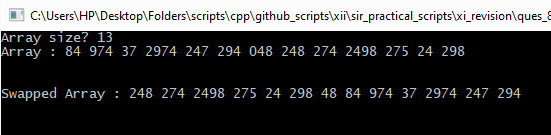
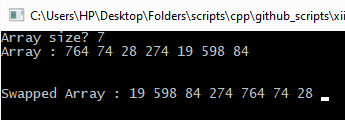
while (getchar() != '\n');

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] << ' ';

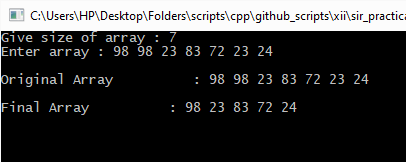
cout << '\n';

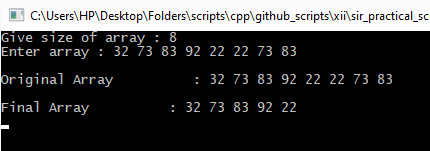
while(getchar() != '\n');

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';

}

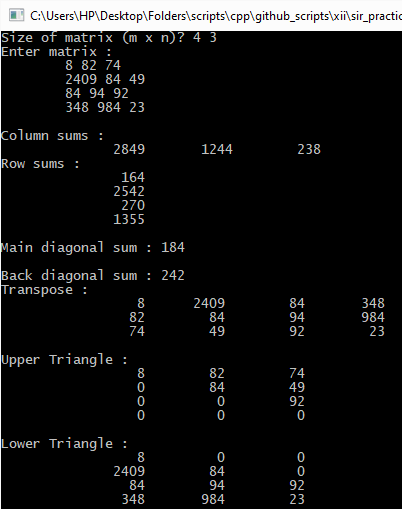
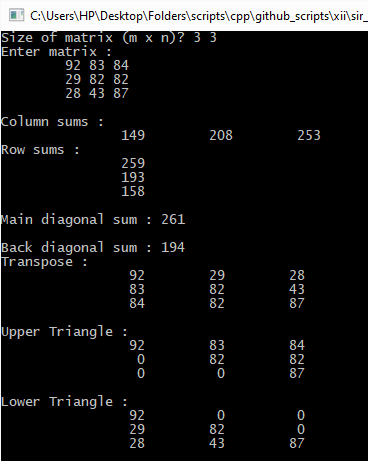
while (getchar() != '\n');

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';

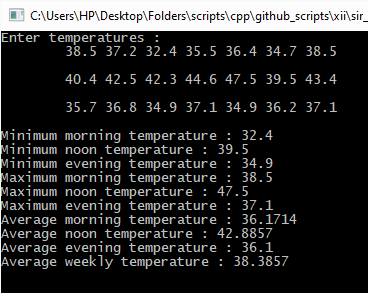
cout << "Average weekly temperature : " << av\_temp\_week << '\n';

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;

}

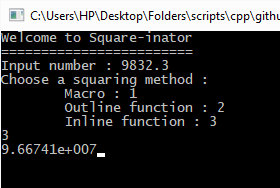
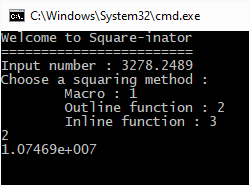
while (getchar() != '\n');

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] << " ";

}

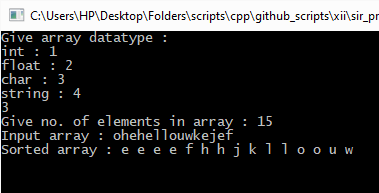
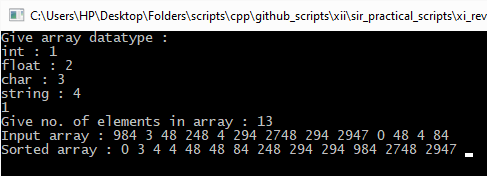
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. Fibonacci

#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');

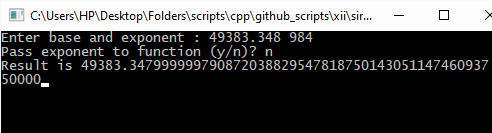
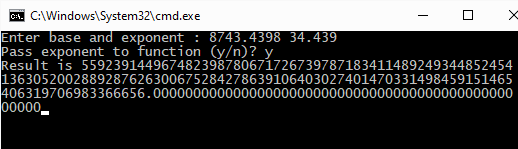
cin.clear();

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;

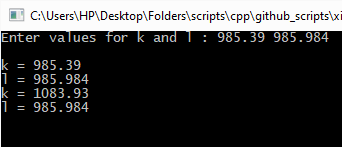
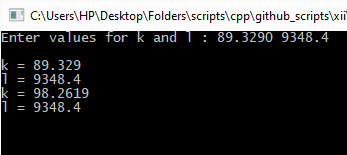
while(getchar() != '\n');

getchar();

return 0;

}

Output :



1. 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 :