**Problem No: 01**

**Topic: Basic Structures: Sets, Functions**

**Problem Title:**

Given two finite sets, list all elements in the Cartesian product of these two sets.

**Objectives:**

To learn Cartesian product of two set.

**Theory:**

A = {x1, x2, x3}

B = {y1, y2, y3}

Cartesian Product: A X B = {(x1, y1), (x2, y2), (x3, y3), (x2, y1), (x2, y2), (x2, y3), (x3, y1), (x3, y2), (x3, y3)}

**Source Code:**

#include <iostream>

using namespace std;

int main(){

int na, nb, i, j;

cout << "Enter no. of elements of set A: "; cin >> na;

cout << "Enter no. of elements of set B: "; cin >> nb;

char a[na], b[nb];

cout << "Enter elements of set A: ";

for(i = 0; i < na; i++) cin >> a[i];

cout << "Enter elements of set B: ";

for(i = 0; i < nb; i++) cin >> b[i];

cout << "\nCartesian product of set A and B:\n";

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

for(j = 0; j < nb; j++){

if(i == na - 1 && j == nb - 1)

cout << "(" << a[i] << ", " << b[j] << ")\n";

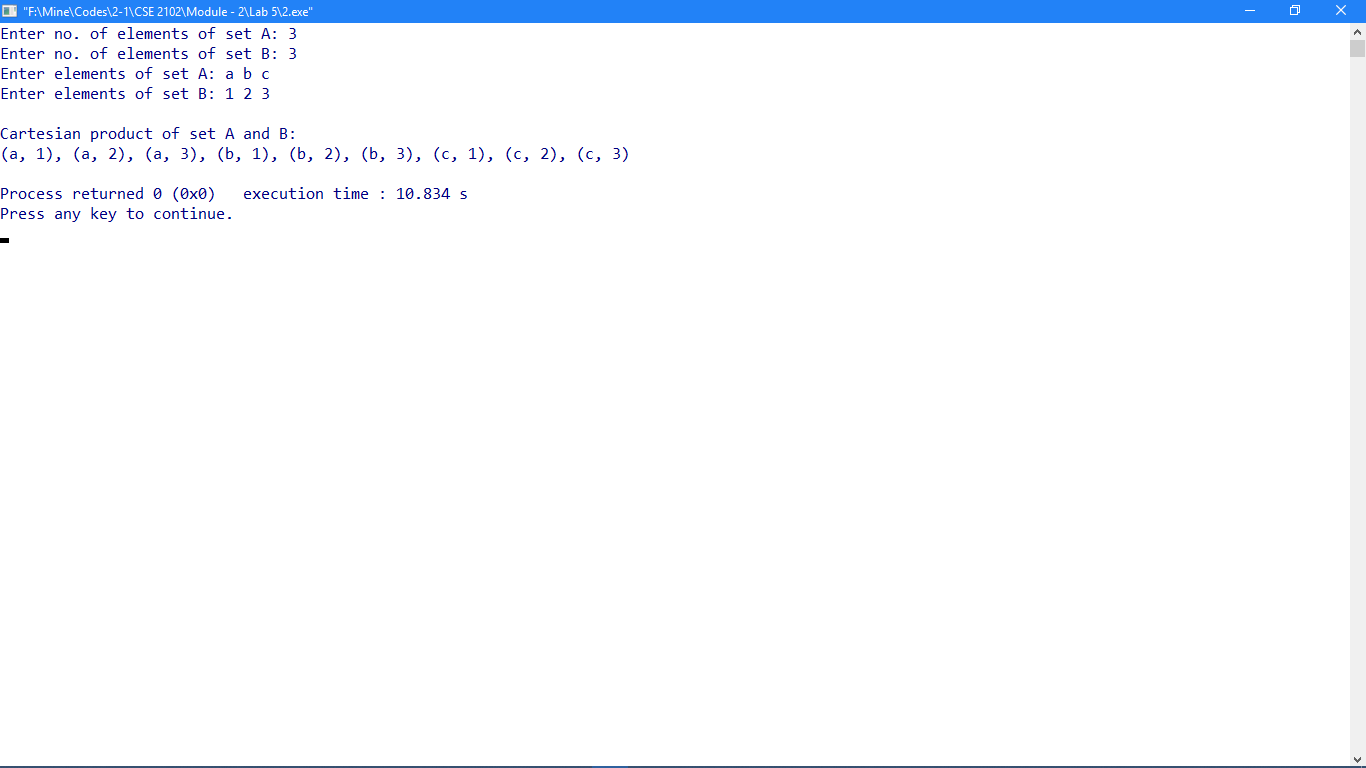
else

cout << "(" << a[i] << ", " << b[j] << "), ";

}

}

**Output:**



**Problem No: 02**

**Topic: Basic Structures: Sets, Functions**

**Problem Title:**

Given a finite set, list all elements of its power set.

**Objectives:**

To learn Power sets of a set.

**Theory:**

A = {a, b, c}

Power set of A is the set of all the subsets of set A.

Power set = {{}, {a}, {b}, {c}, {a, b}, {b, c}, {c, a}, {a, b, c}}

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

class power\_set

{

public:

void printPowerSet(char \*s, int s\_size){

unsigned int ps\_size = pow(2, s\_size);

int j, counter;

for(counter = 0; counter < ps\_size; counter++){

for(j = 0; j < s\_size; j++){

if(counter & (1 << j))

cout << s[j];

}

cout << endl;

}

}

};

int main()

{

power\_set p;

int n;

cout << "Enter no. of elements of set: ";

cin >> n;

char s[n];

cout << "Enter elements of set: ";

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

cin >> s[i];

cout << "Power set elements:\n";

p.printPowerSet(s, n);

return 0;

}

**Output:**



**Problem No: 03**

**Topic: Basic Structures: Sets, Functions**

**Problem Title:**

Given a function f from {1, 2,...,n} to the set of integers, determine whether

i) f(x) = x^2 ii) f(x) = x + 1 iii) f(x) = x^3 + x^2 + x + 1 are one-to-one.

**Objectives:**

To learn one-to-one function.

**Theory:**

A function for which every element of the range of the function corresponds to exactly one element of the domain.

A function f:A->B is 1-1 (or injective) if for each y in B there is at most one x in A such that f(x) = y. Definition 2: A function f:A->B is 1-1 if and only if for every x and y in A, if f(x) = f(y) then x = y.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

int func1(int x){return x \* x;}

int func2(int x){return x + 1;}

int func3(int x){return (x \* x \* x) + (x \* x) + x + 1;}

void check\_func(int \*f);

int n;

int main()

{

int i, j;

cout << "Enter n: ";

cin >> n;

cout << endl;

int x[n], f1[n], f2[n], f3[n];

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

x[i] = i + 1;

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

f1[i] = func1(x[i]);

f2[i] = func2(x[i]);

f3[i] = func3(x[i]);

}

cout << "f(x) = x^2 is ";

check\_func(f1);

cout << "f(x) = x + 1 is ";

check\_func(f2);

cout << "f(x) = x^3 + x^2 + x + 1 is ";

check\_func(f3);

}

void check\_func(int \*f)

{

int flag = 0;

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

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

if(f[i] == f[j]){

flag == 1;break;

}

if(flag == 1)break;

}

}

if(flag == 0)

cout << "One-to-One Function\n";

else

cout << "Not One-to-One Function\n";

}

**Output:**

