8 June

**C Program to Generate All the Set Partitions of n Numbers Beginning from 1 and so on.**

#include<iostream>

using namespace std;

// A utility function to print an array p[] of size 'n'

void printArray(int p[], int n)

{

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

cout << p[i] << " ";

cout << endl;

}

void printAllUniqueParts(int n)

{

int p[n]; // An array to store a partition

int k = 0; // Index of last element in a partition

p[k] = n; // Initialize first partition as number itself

// This loop first prints current partition then generates next

// partition. The loop stops when the current partition has all 1s

while (true)

{

// print current partition

printArray(p, k+1);

// Generate next partition

// Find the rightmost non-one value in p[]. Also, update the

// rem\_val so that we know how much value can be accommodated

int rem\_val = 0;

while (k >= 0 && p[k] == 1)

{

rem\_val += p[k];

k--;

}

// if k < 0, all the values are 1 so there are no more partitions

if (k < 0) return;

// Decrease the p[k] found above and adjust the rem\_val

p[k]--;

rem\_val++;

// If rem\_val is more, then the sorted order is violated. Divide

// rem\_val in different values of size p[k] and copy these values at

// different positions after p[k]

while (rem\_val > p[k])

{

p[k+1] = p[k];

rem\_val = rem\_val - p[k];

k++;

}

// Copy rem\_val to next position and increment position

p[k+1] = rem\_val;

k++;

}

}

// Driver program to test above functions

int main()

{

cout << "All Unique Partitions of 2 \n";

printAllUniqueParts(2);

cout << "\nAll Unique Partitions of 3 \n";

printAllUniqueParts(3);

cout << "\nAll Unique Partitions of 4 \n";

printAllUniqueParts(4);

return 0;

}