Q22. Factorial of large number

i/p → 5

o/p → 120

Now the question comes to our mind that why we need to do this question. If we try to find the factorial of say 50, it is a very large number which can not be stored in pre defined datatypes.
We can do one thing, that simply make an array and store each digit at a particular index and this we can do easily.

Subpart → Add 2 numbers represented in the array
If we do the above subpart, then we can use the same concept in finding the factorial of the number also.

i/p →

9	5	4	9
0	2	1	4

↳ assumed

Here we will be playing with 2 pointer approach.

i → last index of 1st array
j → last index of 2nd array

initially carry = 0
int x = a[i] + b[j] + carry
int digit = x % 10

Push digit to the output array.
carry = x/10;

(i) a[i] = 9
b[j] = 4
x = 9 + 4 + 0 = 13
digit = 13 % 10 = 3 } Push in output
carry = 13/10 = 1

(ii) a[i] = 4
b[j] = 1
x = 4 + 1 + 1 = 6
digit = 6 % 10 = 6 } Push in output
carry = 6/10 = 0

(iii) a[i] = 5
b[j] = 2
x = 5 + 2 + 0 = 7
digit = 7 % 10 = 7 } Push in output
carry = 7/10 = 0

(iv) a[i] = 9
b[j] = 0
x = 9 + 0 + 0 = 9
digit = 9 % 10 = 9 } Push in output
carry = 9/10 = 0

o/p → 3 6 7 9 . So the final answer will be reverse of it & hence 9763 is the output.

Code

```
string calc_sum (int *a, int n, int *b, int m)
{
    //Start from one's place
    int i = n-1;
    int j = m-1;
    string ans;
    //Initially carry is 0.
    int carry = 0;
    //Both arrays are of same length
    while (i>=0 && j>=0) {
        //Find sum
        int x = a[i] + b[j] + carry;
        //Find digit
        int digit = x % 10;
        //Push digit to ans
        ans.push_back (digit + '0');
        //Find carry
        carry = x/10;
        //Update i & j
        i--;
        j--;
    }
    //When a array is bigger
    while (i>=0) {
        //Sum
        int x = a[i] + carry;
        //Digit
        int digit = x % 10;
        //Push
        ans.push_back (digit + '0');
        //Carry
        carry = x/10;
```

```
        // Update
        i--;
    }
    // When b array is bigger
    while (j >= 0) {
        //Sum
        int x = b[j] + carry;
        // Digit
        int digit = x % 10;
        // Push
        ans.push_back (digit + '0');
        // carry
        carry = x / 10;
        // Update
        j--;
    }
    // Carry case handle
    if (carry) {
        ans.push_back (carry + '0');
    }
    // If end digits are zero then remove as when
    the would be reversed, no significance
    while (ans [ans.size() - 1] == '0') {
        ans.pop_back();
    }
    // Reverse
    reverse (ans.begin(), ans.end());
    return ans;
}
```

Time complexity = $O(m+n)$

Space complexity = $O(m+n)$

Factorial of large number

$7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7$

vector / array

ans → | | 1 | ← initially set to 1

(i) $i = 2$

int x = ans[j] * i + carry ;

ans[j] = x % 10 ;

carry = x / 10 ;

$x = 1 \times 2 + 0 = 2$

ans[j] = 2 % 10 = 2

carry = 2 / 10 = 0

ans → | | 2 |

(ii) $i = 3$

$x = 2 \times 3 + 0 = 6$

ans[j] = 6 % 10 = 6

carry = 6 / 10 = 0

ans → | | 6 |

(iii) $i = 4$

$x = 6 \times 4 + 0 = 24$

ans[j] = 24 % 10 = 4

carry = 24 / 10 = 2

ans → | | 2 | 4 |

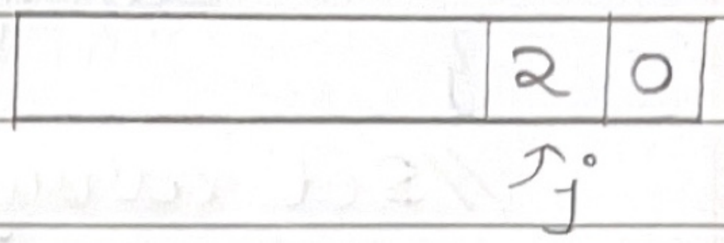
if (carry) → then push

(iv) $i^{\circ} = 5$

$x = 4 \times 5 + 0 = 20$

$ans[j] = 20 \% 10 = 0$

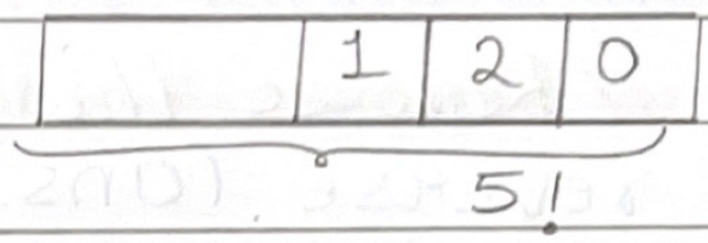
$carry = 20 / 10 = 2$



$x = 2 \times 5 + 2 = 12$

$ans[j] = 12 \% 10 = 2$

$carry = 12 / 10 = 1$



(v) $i = 6$ and then $i = 7$. Same game we can play here.

Code

```
vector <int> factorial (int n) {
        vector <int> ans;
        // Initially push 1 to the ans
        ans.push_back(1);
        int carry = 0;
        // For factorial multiplication
        for (int i = 2 ; i <= n; i++) {
            // Storing answer
            for (int j = 0; j < ans.size(); j++){
                // multiplication
                int x = arr[j] * i + carry;
                // store in ans
                ans[j] = x % 10;
                // Find carry
                carry = x / 10;
        }
        // Push carry
```

```
while (carry) {
        //Store carry digit wise if too big
        ans.push_back (carry %10);
        carry = carry /10;
}
    //Set carry to 0 again
    carry = 0; //Not necessary
}
    // Reverse the ans → final answer
    reverse (ans.begin(), ans.end());
    return ans;
}
```