Handout 2
C++ Programming
Deadline is
October 10

1.

1) Write a function that computes the value of the binomial coefficient $C(n,r)$ *or* $\binom{n}{r}$

$$C(n,r) = \frac{n!}{(n-r)!r!}$$

$$n! = n * (n-1) * (n-2) * \cdots * 1$$

2) Embed your function into a little program that reads two integers **`n`** and **`r`** from **`std::cin`** and writes the value of the binomial coefficient to **`std::cout`**

```cpp
#include <iostream>

// 1) function: compute the value of th binomial coefficient C(n,r)
int combination(int n, int r) {
    if (n < r) return 0;
    if (!r || n == r) return 1;
    return combination(n - 1, r) + combination(n - 1, r - 1);
}
// 2) little program
int main() {
    int n, r;
    std::cin >> n >> r;
    std::cout << combination(n,r);
}
```

2.
Write a function **`permutNumbers`** that prints all n! many permutations of the numbers 1 to n on std::out.

Example: the output for **`permutNumbers(3)`** shall be:
123, 132, 213, 231, 312, 321

```
#include <iostream>

template <class T>
void swap(T &a, T &b) {
    T temp = a;
    a = b;
    b = temp;
}

void print(int seq[], int n) {
    for(int i=0; i<n; i++)
        std::cout << seq[i];
}

void permute(int seq[], int start, int end, int n) {
    if (start == end) {
        print(seq, n);
        puts("");
    } else {
        for (int i = start; i <= end; i++) {
            swap(seq[start], seq[i]);
            permute(seq, start + 1, end, n);
            swap(seq[start], seq[i]); // for backtracking
        }
    }
}

void permutNumbers(int n) {
    int arr[n];
    for (int i=1; i <= n; i++)
        arr[i-1] = i;
    permute(arr, 0, n-1, n);
}

int main() {
    int n;
    std::cin >> n;
    permutNumbers(n);
}
```

3. Given the following function definition:

```
int sum_down(int x)
{
     if (x >= 0)
     {
          x = x - 1;
          int y = x + sum_down(x);
          return y + sum_down(x);
     }
     else
     {
          return 1;
     }
}
```

a) What is this smallest integer value of the parameter **x**, so that the returned value is greater than 1.000.000 ?

x = 19, sum_down(19) = 1048556

b) Rewrite the function, so that it is free of recursion. I.e. give an iterative definition on the foundation of a loop-construct.
TIP: First transform the recursive definition, so that you have just a single recursive call.

```
int sum_down_no_recursion(int x)
{
    if (x <= 0) return 1;
    int current = 1, next = 0;
    for (int i = 0; i < x; i++)
    {
        next = 2 * current + i;
        current = next;
    }
    return current;
}
```

c) Is it OK to switch the type of the parameter **x** to **double**?
Discuss your decision / give an argumentation.

It's OK. In the given function, there are only addition or substract operations, so it can compute the result successfully whether x is int or double.

d) Is it OK to switch the type of the parameter **x** to **unsigned int**? Discuss your decision / give an argumentation.

It's OK. In the given function or function in b), It only computes when the x is positive number. Otherwise, just return 1. So switching the type of x to unsigned int, it can still compute.

e) Is it OK to switch the function head to **int sum_down(const int x)**? Discuss your decision / give an argumentation.

No. 'const', the reserved keyword in C++ makes the variable immutable. In the given function, it substracts some integer from the parameter x. But it is okay in the function in b), since that x never changed.