1. N-й факториал F(n) для натуральных чисел n определяется следующим образом:
   F(0) = 1
   F(n) = 1*2*3*...*(n-1)*n

Так:
   F(2) = 1*2 = 2
   F(3) = 1*2*3 = 6
   и т.д.

Необходимо:
   a) Написать итеративную (нерекурсивную) функцию для нахождения F(n)
   b) Написать рекурсивную функцию для нахождения F(n)
   c) Объяснить какая версия будет быстрее и почему?

2. Реализовать функцию ReverseAfter() для обращения односвязного списка от первого вхождения заданного значения.
Например, если подать на вход A B C D E F и значение D, функция возвращает A B C F E D.
Реализация должна быть inplace, т.е. не создавать дополнительных узлов.

```
struct Node
{
   struct Node* next;
   int val;
};

ReverseAfter( struct Node* head, int val );
```

3. Найти как можно больше ошибок в следующем коде.

```
// Function to copy 'nBytes' of data from src to dst.
void myMemcpy(char* dst, const char* src, int nBytes)
{
   // Try to be fast and copy a word at a time instead of byte by byte
   int* wordDst = (int*)dst;
   int* wordSrc = (int*)src;
   int numWords = nBytes >> 2;
   for (int i = 0; i < numWords; i++)
   {
      *wordDst++ = *wordSrc++;
   }

   int numRemaining = nBytes - (numWords << 2);
   dst = (char*)wordDst;
   src = (char*)wordSrc;
   for (int i = 0; i <= numRemaining; i++);
   {
      *dst++ = *src++;
   }
}
```

Примечание: в задании содержатся (как минимум)
   1 алгоритмическая ошибка
   2 проблемы с переносимостью
   1 синтаксическая ошибка

4. Написать функцию, удаляющую последовательно дублирующиеся символы в строке.
Например:
In: "AAA BBB AAA"
Out: "A B A"

```
void RemoveDups( char *pStr );
```

Примечание: реализация не должна содержать библиотечных классов и функций, которые могут являться ре шением данного задания.

5. Реализовать функции сериализации и десериализации двусвязного списка в бинарном формате в файл.

```
struct ListNode {
    ListNode* prev;
    ListNode* next;
    ListNode* rand; // указатель на произвольный элемент данного списка либо NULL
    std::string data;
};

class List {
public:
    void Serialize   (FILE* file);  // сохранение в файл (файл открыт с помощью fopen(path, "wb"))
    void Deserialize (FILE* file);  // загрузка из файла (файл открыт с помощью fopen(path, "rb"))

private:
    ListNode* head;
    ListNode* tail;
    int count;
};
```

Примечание: сериализация подразумевает сохранение и восстановление полной структуры списка, включая взаимное соотношение его элементов между собой.