Contents

1 Libraries

1.1 cstdlib

```cpp
#include <cstdlib>
using namespace std;

int main() {

    // Functions
        // String conversion
            double atof (const char* str);  // Convert string to double; return 0.0 if no conversion
            int atoi (const char * str);    // Convert string to integer;
            long int atol ( const char * str ); // Convert string to long integer; return 0 if no conversion
            long long int atoll ( const char * str ); // Convert string to long long integer; return 0 if no conversion
            double strtod (const char* str, char** endptr); // Convert string to double; return 0.0 if no conversion, HUGE_VAL(cmath) if out of range
            float strtof (const char* str, char** endptr);  // Convert string to float

strtol
    Convert string to long integer (function)

strtold
    Convert string to long double (function)

strtoll
    Convert string to long long integer (function)

strtoul
    Convert string to unsigned long integer (function)

strtoull
    Convert string to unsigned long long integer (function)


Pseudo-random sequence generation

rand
    Generate random number (function)

srand
    Initialize random number generator (function)


Searching and sorting

bsearch
    Binary search in array (function)

qsort
    Sort elements of array (function)


Integer arithmetics

abs
    Absolute value (function)

div
    Integral division (function)

labs
    Absolute value (function)

ldiv
    Integral division (function)

llabs
    Absolute value (function)

lldiv
    Integral division (function)



Macro constants

NULL
    Null pointer (macro)

RAND_MAX
    Maximum value returned by rand (macro)



Types

div_t
    Structure returned by div (type)

ldiv_t
    Structure returned by ldiv (type)

lldiv_t
    Structure returned by lldiv (type)

size_t
    Unsigned integral type (type)


    return 0;
}
```

1.2 algorithm

```cpp
#include <algorithm>
using namespace std;

int main() {

    void sort (RandomAccessIterator first, RandomAccessIterator last);
    /*  Sorts the elements in the range [first,last) into ascending order
     *  In N*lg(N) complexity
     */

    ForwardIterator lower_bound (ForwardIterator first, ForwardIterator last, const T& val);
    /*  Returns an iterator pointing to the first element in the range [first,last) which is >= val
```

```
13      *  In lg(N)+1 complexity
14      *      requires sorted elements
15      */
16
17      ForwardIterator upper_bound (ForwardIterator
            first, ForwardIterator last, const T& val);
18      /*  Returns an iterator pointing to the first
            element in the range [first,last) which is >
            val
19      *  In lg(N)+1 complexity
20      *      requires sorted elements
21      */
22
23      pair<ForwardIterator,ForwardIterator> equal_range
            (ForwardIterator first, ForwardIterator last,
            const T& val);
24      /*  Returns the bounds of the subrange with all
            the elements == val of the range [first,last)
25      *      return type equivalent to pair<
             lower_bound(), upper_bound>
26      *  In 2*lg(N)+1 complexity
27      *      requires sorted elements
28      */
29
30      bool next_permutation (BidirectionalIterator
            first, BidirectionalIterator last);
31      /*  Rearranges the elements in the range
            [first,last) into the next lexicographically
            greater permutation, then returns
32      *      true if could rearrange as a
             lexicographicaly greater permutation
33      *      false if no greater arrangement than the
             previous (and sorted in ascending order)
34      *  In N/2 complexity
35      */
36
37      bool prev_permutation (BidirectionalIterator
            first, BidirectionalIterator last);
38      /*  Rearranges the elements in the range
            [first,last) into the previous
            lexicographically-ordered permutation, then
            returns
39      *      true if could rearrange as a
             lexicographicaly smaller permutation
40      *      false if arrangement is the largest
             possible (and sorted in descending order)
41      *  In N/2 complexity
42      */
43
44      return 0;
45  }
```


- 中文測試
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$