Contents

1 section1 1
1.1 basic 1
1.2 test 1

2 Libraries 1
2.1 algorithm 1

3 Section2 1
3.1 thm 1

1 section1

1.1 basic

```cpp
// c++ code
#include <bits/stdc++.h>
using namespace std;

int main() {
    // test comment
    cout << "test string\n";
}
```

1.2 test

```cpp
// map::begin/end
#include <iostream>
#include <map>

using namespace std;

int main ()
{
  map<char,int> mymap;

  mymap['b'] = 100;
  mymap['a'] = 200;
  mymap['c'] = 300;

  // show content:
  for (map<char,int>::iterator it = mymap.begin(); it
      != mymap.end(); ++it)
    cout << it->first << " => " << it->second << '\n';

  return 0;
}
```

2 Libraries

2.1 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
     *  In lg(N)+1 complexity
     *      requires sorted elements
     */

    ForwardIterator upper_bound (ForwardIterator
        first, ForwardIterator last, const T& val);
    /*  Returns an iterator pointing to the first
        element in the range [first,last) which is >
        val
     *  In lg(N)+1 complexity
     *      requires sorted elements
     */

    pair<ForwardIterator,ForwardIterator> equal_range
        (ForwardIterator first, ForwardIterator last,
        const T& val);
    /*  Returns the bounds of the subrange with all
        the elements == val of the range [first,last)
     *      return type equivalent to pair<
         lower_bound(), upper_bound>
     *  In 2*lg(N)+1 complexity
     *      requires sorted elements
     */

    bool next_permutation (BidirectionalIterator
        first, BidirectionalIterator last);
    /*  Rearranges the elements in the range
        [first,last) into the next lexicographically
        greater permutation, then returns
     *      true if could rearrange as a
         lexicographicaly greater permutation
     *      false if no greater arrangement than the
         previous (and sorted in ascending order)
     *  In N/2 complexity
     */

    bool prev_permutation (BidirectionalIterator
        first, BidirectionalIterator last);
    /*  Rearranges the elements in the range
        [first,last) into the previous
        lexicographically-ordered permutation, then
        returns
     *      true if could rearrange as a
         lexicographicaly smaller permutation
     *      false if arrangement is the largest
         possible (and sorted in descending order)
     *  In N/2 complexity
    */

    return 0;
}
```

3 Section2

3.1 thm

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