**CPP Problem Design**

|  |
| --- |
| **Subject: Adding Large Numbers** |
| **Contributor: 謝宜杭,** **林承達, 廖宣瑋** |
| **Main testing concept: String Processing**   |  |  | | --- | --- | | **Basics** | **Functions** | | □ C++ BASICS  □ FLOW OF CONTROL  □ FUNCTION BASICS  □ PARAMETERS AND OVERLOADING  □ ARRAYS  □ STRUCTURES AND CLASSES  □ CONSTRUCTORS AND OTHER TOOLS  □ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES  ■ STRINGS  □ POINTERS AND DYNAMIC ARRAYS | □ SEPARATE COMPILATION AND NAMESPACES  □ STREAMS AND FILE I/O  □ RECURSION  □ INHERITANCE  □ POLYMORPHISM AND VIRTUAL FUNCTIONS  □ TEMPLATES  □ LINKED DATA STRUCTURES  □ EXCEPTION HANDLING  □ STANDARD TEMPLATE LIBRARY  □ PATTERNS AND UML | |
| **Description:**  Given two integers A, B. Please calculate the adding result of A+B, notice that the maximum bits of the number can approximate to 10000.  Don’t try to use **long long**, **long long int**, or **\_m128**, etc. These variable types are invalid because the maximum bits of the given number will approximate to 10000((10^10000)-1).  Make sure the input number is valid, or print “**Not a valid number, please try again.**”  Please package the big number as a **structure** (e.g. struct BigInt…) with a simple object type for each bit.  To add the two big numbers, please use the "function Add(...)" shown as below.  P.S. **const&** won’t affect the grammar of parameter passing but can avoid unnecessary memory usage. Use it or not depends on you.  Struct BigInt  {  …  }  BigInt Add(const BigInt &lhs,const BigInt &rhs)  {  //Calculation  Return …;  }  int main()  {  BigInt a,b;  …Input a,b  BigInt result = Add(a,b);  …Output result  }  **Input:**  The first line of input contains an integer **N** (100 > N > 0), which indicates there’re N following data pairs. Note that every two lines makes a data pair, and each pair contains two big integers A and B on a line by itself. The maximum bits of A, B is（10^l0000) – 1.  **Output:**  Print the result of A+B.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | 3  43789507384925798320000000000000000000000000001  44997439848794037580000000000000000000000000002  1bbbba45  1234567  1  9 | 8878694723371983590000000000000000000000000003  Not a valid number, please try again.  10 | |
| **□ Eazy,Only basic programming syntax and structure are required.**  **■ Medium,Multiple programming grammars and structures are required.**  **□ Hard,Need to use multiple program structures or more complex data types.** |
| **Expected solving time:**  25 minutes |
| **Other notes:** |