**10802 CPP Final Exam**

|  |
| --- |
| **Subject: N Dim Vector** |
| **Contributor: 陳泳峰、陳宥潤和范茗翔** |
| **Main testing concept: Class Implementation**   |  |  | | --- | --- | | **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:**  Please define a class, named **VecNf,** to store a multidimensional vector along with several vector operations. The required implementations are listed in the following.   * Class **VecNf** has three constructors: * **VecNf(void):** Constructs a one-dimensional zero vector in default. * **VecNf(float \*dest, int cnt)**: Constructs a cnt dimensional vector using the first cnt numbers in dest. * **VecNf(const VecNf& rhs):** Constructs a new vector from another multidimensional vector of rhs. * Class **VecNf** has a member function **int Size()**, which return the dimension of VecNf. * You are required to overload the following operators to meet their respective operational requirements. Here, we use two 2D vectors, v1(2,2) and v2(6,1) as examples. * **Assignment(defines operator =):** Assigns a **VecNf** to another **VecNf**, and at the same time, output a message of "ASSIGNMENT!!!". For example, vec = v1; →vec is (2, 2). * **Indexing(defines operator []): Returns the value** of the certain dimension variable in the vector. For example, v1[0] needs to return the value at dimension 0, which has the value of 2. If the index is larger than the dimension, output a message of "Out of range." and return 0. * **Addition(defines operator +):**Addstwo vectors, and at the same time, output a message of "ADDITION!!!". For example, vec = v1 + v2; →vec is(8, 3). * **Subtraction(defines operator -):** Subtracts one vector from another, and at the same time, output a message of "SUBTRACTION!!!". For example, vec = v1 - v2; →vec is(-4, 1). * **Multiplication(defines operator \*):** Computes the inner product of two vectors, and at the same time, output a message of "INNER PRODUCT!!!". For example, float tmp = v1 \* v2; →tmp is 14. * **Multiplication(defines operator \*):** Computes the scalar multiplication of a floating-point number and a vector, and at the same time, output a message of "SCALAR!!!". For example, vec = 5.0f \* v1; →vec is (10, 10). * **Output Stream(defines operator <<):** Outputs the vector. For example, cout << v1; → Print out the message of “(2,2)”.   Note:   1. If the index is out of range, please output a message of “Out of range.”, and return 0. 2. For any arithmetic operator, the two **VecNf** must have the same dimensional. If they are not, please output the message of "Dimensions inconsistent" and return 0 vector or 0.   **Input:**  No inputs.  Note:   1. The main() function in your submission will be replaced when judging. 2. You can implement **VecNf** class, in other file(exclude main.cpp).   **Output:**  The result of executing your program with the given main function.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | No inputs. | ASSIGNMENT!!!  3 2 Out of range.  0  ASSIGNMENT!!!  1 2 3  ADDITION!!!  ASSIGNMENT!!!  7 7 7  4  INNER PRODUCT!!!  28  SCALAR!!!  (15,10)  Dimensions inconsistent  0  (6,5,4) | |
| **■ Easy. Only basic programming syntax and structure are required.**  **□ Medium. Multiple programming grammars and structures are required.**  **□ Hard. Need to use multiple program structures or complex data types.** |
| **Expected solving time:**  40 minutes |
| **LTE:**  1Sec |
| **Other notes:**  #include "VecNf.h"  void doNothing(VecNf tar) { return; }  int main()  {  float a\_value[] = { 3.0, 2.0 };  float b\_value[] = { 1, 2, 3 };  float c\_value[] = { 6, 5, 4 };  VecNf A(a\_value, 2);  VecNf B(b\_value, 3);  VecNf C(c\_value, 3);  VecNf T;  T = A; // Assignment  for (int i = 0; i < T.Size() + 1; i++) {  cout << T[i] << " ";  } cout << endl;  T = B; // Assignment  for (int i = 0; i < T.Size(); i++) {  cout << T[i] << " ";  } cout << endl;  T = B + C; // Vector addition  for (int i = 0; i < T.Size(); i++) {  cout << T[i] << " ";  } cout << endl;  doNothing(T); // call by value  cout << C[2] << endl; // Indexing  cout << C \* B << endl; // Inner Product  cout << 5 \* A << endl; // Scalar  cout << A \* C << endl; // Inconsistent  cout << C << endl; // Outputs the vector    return 0;  } |