**CPP Problem Design**

|  |
| --- |
| **Subject: Dice** |
| **Contributor: 謝宜杭,** **謝公耀, 廖宣瑋** |
| **Main testing concept: Polymorphism**   |  |  | | --- | --- | | **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:**  Listed below is a Dice class that simulates rolling a die with a different number of sides. The default is a standard die with six sides. The rollTwoDice function simulates rolling two dice objects and returns the sum of their values. The srand function requires including cstdlib and time.h.  class Dice  {  public:  Dice();  Dice(int numSides);  virtual int rollDice() const;  protected:  int numSides;  };  Dice::Dice()  {  numSides = 6;  srand(time(NULL)); *// Seeds random number generator*  }  Dice::Dice(int numSides)  {  this->numSides = numSides;  srand(time(NULL)); *// Seeds random number generator*  }  int Dice::rollDice() const  return (rand() % numSides) + 1;  }  *// Take two dice objects, roll them, and return the sum*  int rollTwoDice(const Dice& die1, const Dice& die2)  {  return die1.rollDice() + die2.rollDice();  }  Create your own class, LoadedDice,that is derived from Dice. Add a default constructor and a constructor that takes the number of sides as input. Override the rollDice function so that with a 50% chance the function returns the largest number possible(i.e. numSides) and with a 50% chance return what Dice’s rollDice function returns.  Note that please use this following code snippets as your main()  //Main  int main()  {  //Uncomment the line below for regular dice  Dice die1(6), die2(6);  LoadedDice die3(6), die4(6);  // This would be the game; here we just simulate it rolling 10 times  for (int i = 0; i < 10; i++)  {  int total = rollTwoDice(die1, die2);  cout << total << " ";  }  cout << endl;  for (int i = 0; i < 10; i++)  {  int total = rollTwoDice(die3, die4);  cout << total << " ";  }  cout << endl;  return 0;  }  **Input:**  **Output:**  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | NaN | Random Output | |
| **□ 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:**  30 minutes |
| **Other notes:** |