**CPP Problem Design Example**

|  |
| --- |
| **Subject: Polygon class** |
| **Contributor: 林承達, 楊晉復, 廖宣瑋** |
| **Main testing concept: Polymorphism, Stack unwinding**   |  |  | | --- | --- | | **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 use concept of polymorphism to design classes as follow:    These classes provide functions for the user to calculate area and sides length of some polygons. (e.g., circle, triangle, rectangle, hexagon.)  The sides array is defined as follow (All the input of side must be integer!  ):   1. Circle, the size of the sides array is 1, which stores its radius. 2. Triangle, the size of the sides array is 3, which stores 3 sides of triangle. 3. Rectangle, the size of the sides array is 4, which stores 4 sides of rectangle in length, width, length, width order. 4. Hexagon, the size of the sides array is 6, which stores 6 sides of   hexagon. (Length of all 6 sides are the same.)  Also, you must implement 3 virtual functions, namely area, sideLength and sideCheck in each child class for the following usage:   1. int area():Return the area of the polygon in integer.(Round down to an integer) 2. int sideLength():Return the perimeter of the polygon in integer. (Round down to an integer) 3. void sideCheck():Check the input integers of the sides, if the input integers contain negative integer, then throw exception "Side length cannot be negative or zero."; if the format of input is wrong, then throw exception "Invalid side.”. You need to release the memory of sides when exception occurred.   printArea and printSideLength only output the calculated polygon area and side length. See the sample output for the output format.  In addition, the concept of RAII needs to be applied, so the memory resources can be correctly released when the exception is triggered.  **To validate the implementation of RAII, please declare an integer pointer called ptr in main.cpp and point to the starting address of sides. If the memory is not properly released, console will output "Wrong." If the polygon is correctly created, console will output "Correct.". We will replace main.cpp to test the program you wrote.**  **Input:**  Please see the sample main.cpp  **Output:**  Output the area and perimeter of polygon in integer.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | int main()  {  int sc = 3;  Polygon\* poly = NULL;  int \*arr = NULL;  int \*ptr = NULL;  try  {  cout << "Triangle" << "\n";  arr = new int[sc];  for (int i = 0; i < sc; i++)  arr[i] = -6;  ptr = arr;  poly = new Triangle(arr, sc);  poly->sideCheck();  poly->printArea();  poly->printSideLength();  }  catch (const char\* errorMsg)  {  if (\*ptr == -6)cout <<"Wrong."<<"\n";  cerr << errorMsg << "\n";  }    try  {  cout << "Rectangle" << "\n";  sc = 4;  arr = new int[sc];  for (int i = 0; i < sc; i++)  arr[i] = 4;  ptr = arr;  poly = new Rectangle(arr, sc);  poly->sideCheck();  poly->printArea();  poly->printSideLength();  }  catch (const char\* errorMsg)  {  cerr << errorMsg << "\n";  }  if (\*ptr == 4)cout << "Correct." << "\n";  try  {  cout << "Circle" << "\n";  sc = 1;  arr = new int[sc];  arr[0] = -5;  ptr = arr;  poly = new Circle(arr,sc);  poly->sideCheck();  poly->printArea();  poly->printSideLength();  }  catch (const char\* errorMsg)  {  if (\*ptr == -5)cout << "Wrong." << "\n";  cerr << errorMsg << "\n";  }    try  {  cout << "Hexagon" << "\n";  sc = 6;  arr = new int[sc];  for (int i = 0; i < sc; i++)  arr[i] = 7;  ptr = arr;  poly = new Hexagon(arr, sc);  poly->sideCheck();  poly->printArea();  poly->printSideLength();  }  catch (const char\* errorMsg)  {  cerr << errorMsg << "\n";  }  if (\*ptr == 7)cout << "Correct." << "\n";  return 0;  } | Triangle  Side length cannot be negative or zero.  Rectangle  Area:16  Perimeter:16  Correct.  Circle  Side length cannot be negative or zero.  Hexagon  Area:127  Perimeter:42  Correct. | |
| **□ 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:**  25 minutes |
| **Other notes:**  none |