**BP1000. Hex to decimal**

**Description**

Write a function that parses a hex number as a string into a decimal integer.   
The function header is as follows:  
  
int **parseHex**(const char \* const hexString)  
  
For example, hexString "A5" is 165 (10\*16+5=165). So, parseHex("A5") returns 165.  
Your implementation of parseHex function should throw a **runtime\_error** exception if the binaryString is not a hex string.  
The test function f():  
void f()  
{  
  try  
  {  
    cout << parseHex("10") << endl;  
    cout << parseHex("ABC") << endl;  
    cout << parseHex("9G3") << endl;  
  }  
  catch (runtime\_error ex)  
  {  
    cout << ex.what() << endl;  
  }  
}  
should output:  
16  
2748  
Hex number format error

**BP1001. Binary to decimal**

**Description**

Write a function that parses a binary number as a string into a decimal integer.   
The function header is as follows:  
  
int **parseBinary**(const char \* const binaryString)  
  
Your implementation of parseBinary function should throw a **runtime\_error** exception if the binaryString is not a binary string.  
The test function f():  
void f()  
{  
  try  
  {  
    cout << parseBinary("10001") << endl;  
    cout << parseBinary("11111111") << endl;  
    cout << parseBinary("21111111") << endl;  
  }  
  catch (runtime\_error ex)  
  {  
    cout << ex.what() << endl;  
  }  
}  
should output:  
17  
255  
Binary number format error

**BP1002. Modify Date class**

**Description**

class Date  
{  
public:  
  Date(int y=0, int m=1, int d=1);    
  static bool leapyear(int year);  
  int getYear() const;  
  int getMonth() const;  
  int getDay() const;  
bool operator<(Date &d);  
bool operator<=(Date &d);  
bool operator==(Date &d);  
bool operator!=(Date &d);  
bool operator>(Date &d);  
bool operator>=(Date &d);  
int &operator[](const string &s);  
      
  // add any member you need here    
};  
You implementation should enable the usage like this:  
  
  Date date(2011,4,1);  
  cout << date["year"] << endl; // output 2011  
  cout << date["month"] << endl;// output 4  
  cout << date["day"] << endl; // output 1  
  date["year"] = 2007;  
  date["month"] = 11;  
  date["day"] = 11;  
  cout << date["year"] << endl; // output 2007  
  cout << date["month"] << endl;// output 11  
  cout << date["day"] << endl; // output 11  
  
What to do if the subscript is not one of "year", "month" or "day"?    
Define a custom exception class named IllegalSubscriptException and let the function operator [] throw an **IllegalSubscriptException** if the subscript is not one of "year", "month" or "day".  
The test function f():  
void f()  
{  
  Date date1(2011,4,1);  
  try  
  {  
    cout << date1["abc"] << endl;  
  }  
  catch(IllegalSubscriptException ex)  
  {  
    cout << "Illegal Subscript Exception" << endl;    
  }  
  
  try  
  {  
      date1["abc"] = 2000;  
  }  
  catch(IllegalSubscriptException ex)  
  {  
    cout << "Illegal Subscript Exception" << endl;  
  }  
}  
should output:  
Illegal Subscript Exception  
Illegal Subscript Exception

**BP1003. Modify StackOfIntegers class**

**Description**

class StackOfIntegers  
{  
public:  
  StackOfIntegers();  
  bool empty();  
  int peek();  
  void push(int value);  
  int pop();  
  int getSize();  
};  
Define a custom exception class named EmptyStackException and let the peek and pop function throw an ExmptyStackException if the stack is empty.  
The test function f():  
void f()  
{  
  StackOfIntegers s;  
  
  try  
  {  
    s.peek();  
  }  
  catch(EmptyStackException ex)  
  {  
    cout << ex.what() << endl;  
  }  
    
  try  
  {  
    s.pop();  
  }  
  catch(EmptyStackException ex)  
  {  
    cout << ex.what() << endl;  
  }  
}  
should output:  
Empty Stack Exception  
Empty Stack Exception

**BP1004. 简单减法**

**Description**

某小朋友已经四岁了，她开始学习数字减法，但她还不理解负数的概念，如果被减数和减数中出现负数，或者被减数小于减数，她都不会算。  
函数calc做减法计算前进行了检查。  
int calc(int a, int b) throw(logic\_error)  
{  
  if (a<0) throw out\_of\_range("Out of range exeception");  
  else if (b<0) throw out\_of\_range("Out of range exeception");  
  else if (a<b) throw logic\_error("Minuend smaller than subtrahend");  
  return a-b;  
}  
请写一个test函数  
void test(int, int)  
使得以下函数f  
void f()  
{

  test(3, 1);  
  test(-3,1);  
  test(1,-3);  
  test(1,3);  
}  
输出如下：

2  
Out of range exeception  
Out of range exeception  
Minuend smaller than subtrahend

请只提交test函数，不要提交calc函数。