Creative Software Design, Assignment 12-2

Deadline: 2024-11-19 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format:
  13178_Assignment[Assignment-number]_[Student-ID].zip
  - Ex. 13178_Assignment1-1_2024123456.zip
- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**
- **You must submit your solution in the zip file before the deadline.**

1. Implementing MyVector Class with Exception Handling

A. Task Overview

Recall problem 1 from Assignment 8-2. Implement the MyVector class by inheriting from the provided Container class. The MyVector class should function as a dynamic vector data structure and handle exceptional cases using try-throw-catch blocks.

B. Provided Class Structure

```
class Container {
public:
    virtual void push_back(int value) = 0;
    virtual void pop_back() = 0;
    virtual int getVectorArr(int index) = 0;
    virtual void setVectorArr(int index, int value) = 0;
};
```

C. Requirements

1. **Data Storage**
   i. The data type of `myVector` should be `int`, stored in a dynamic array within `MyVector`.
   ii. **Do not** use any STL containers or external libraries for storage.
2. **Member Functions**
   i. Constructor and Destructor
   ii. `push_back(int value)`
   iii. `pop_back()`
   iv. `getVectorArr(int index)`
   v. `setVectorArr(int index, int value)`
3. The maximum size of `MyVector` should be set by the constructor.
4. If `MyVector` is full when `push_back()` is called, double the size of the dynamic array before adding the new item.
5. If `MyVector` is empty and `pop_back()` is called, throw an exception with the message "`nothing to pop`".

6. **Index Range Validation**

   i. `getVectorArr(int index)`: Returns the element at the specified index. Throw "`Out of range error`" if the index is out of bounds (valid range: `-1 < index < size`)

   ii. `setVectorArr(int index, int value)`: Updates the element at the specified index. Throw "`Out of range error`" if the index is out of bounds (valid range: `-1 < index < size`)

D. **Example of the `main()` function:**

```
int main() {
    Container* v = new MyVector(5);
    for (int i = 0; i < 10; i++) {
        v->push_back(i * 10);
    }
    try {
        v->setVectorArr(11, 20);
        cout << v->getVectorArr(10) << endl;
    } catch (const char* msg) {
        cerr << msg << endl;
    }
    for (int j = 0; j < 11; j++) {
        try {
            v->pop_back();
        } catch (const char* msg) {
            cerr << msg << endl;
        }
    }
    cout << endl;
}
```

E. Example output of your program (Bold text indicates user input):

```
Out of range error
90 80 70 60 50 40 30 20 10 0
nothing to pop
```

F. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. Implementing MyArray and MyException Classes with Exception Handling

A. Task Overview

Implement the MyArray and MyException classes. You must handle exceptional cases using try-throw-catch blocks properly.

B. `MyArray` Class

1. Private Members

i. `int* my_array`: Stores data in a dynamic array.

ii. `int maxSize`: Holds the size of the array.

2. Public Members

i. Constructor: Initializes `maxSize` using an initializer list and allocates a dynamic array of size `maxSize`.

ii. `int& operator[](const int& index)`:

- Provides array-like access using the `[]` operator.
- If `index` is out of bounds, throw a `MyException`.
- Returns `my_array[index]`.

C. `MyException` Class

1. Public Members

i. `void report()`: Outputs the error message "`Exception report`".

D. Example of the `main()` function:

```
int main() {
    MyArray myArray(5);
    try {
        myArray[20];
    }
    catch (MyException & e) {
        e.report();
    }
    return 0;
}
```

E. Example output of your program (Bold text indicates user input):

```
Exception report
```

F. Submission file: one C++ source file (File name: 2.**cc** or 2.**cpp**)

3. Handling Exceptional Cases in Input Processing

A. Task Overview

Recall problem 3 from Assignment 2-2. Previously, it was assumed that the number of input words would be re-entered if they exceeded a specified range. Rewrite the program for problem 3 from Assignment 3-2 without this assumption. Implement exception handling for invalid inputs using try-throw-catch blocks.

B. Problem 3 of Assignment 2-2 Reference Code

```
#include <iostream>
using namespace std;

int checkN() {
    int num;
    while (1) {
        cout << "S = ";
        cin >> num;
        if (num > 0 && num <= 100) {
            break;
        }
    }
    return num;
}

int main(void) {
    int N, cnt = 0;
    string input;
    N = checkN();
    for (int i = 0; i < N; i++) {
        while (true) {
            int eli = 0;
            cin >> input;
            for (int i = 0; i < input.size(); i++) {
                if (!(input[i] >= 'a' && input[i] <= 'z')) {
                    cout << "warning" << endl;
                    eli = 1;
                    break;
                }
            }
            if (eli == 0) { break; }
        }

        bool group = true;
        for (int j = 0; j < input.size(); j++) {
            for (int k = 0; k < j; k++) {
                if (input[j] != input[j - 1] && input[j] == input[k]) {
                    group = false;
                    break;
                }
            }
        }
        if (group == true) {
            cnt++;
        }
    }
    cout << cnt;
    return 0;
}
```

C. **Updated Requirements**

1. **Input Validation**

   i. The program should handle cases where `num` (input value for N) is out of the valid range (1 to 100).

   ii. If `num` is out of range, throw an exception and handle it using try-throw-catch.

D. Example output of your program (Bold text indicates user input):

```
(example 1)
S = 101
Out of range error.
```

```
(example 2)
S = 3
i
like
strawberry
Output: 2
```

```
(example 3)
S = 3
i
lIkE
warning
love
you
Output: 3
```

E. Submission file: one C++ source file (File name: 3.**cc** or 3.**cpp**)