CSE 335 Spring 2016 Exam 1

All answers should be written on the answer sheet. Do not write answers on problem sheet.

Question 1: Answer the following questions, assuming all needed header files are included.

Question 1.1 (5 points.)

Will the following program be able to compile? If yes, what's the output? If no, state your reason.

```
1. int main(int argc, char** argv) {
2.     const int bufferSize = 100;
3.     const int* intPtr=&bufferSize;
4.     cout<<*intPtr<<endl;
5.     return 0;
6. }
```

Answer: yes, 100.

Question 1.2 (5 points.)

Will the following program be able to compile? If yes, what's the output? If no, state your reason.

```
1. class A{
2. protected:
3.     int m_data;
4. public:
5.     int f() const{
6.         m_data++;
7.     }
8.     void print() const{
9.         cout<<m_data<<endl;
10.    }
11. };
```

Answer: not, because f() is a constant member function.

Question 1.3 (5 points.)

List the output produced by the following program?

```
1. class A{
2. public:
3.     virtual void print() const{
4.         cout<<"I am in A"<<endl;
5.     }
6. };
7.
8. class B: public A{
9. public:
10.    void print(){
11.        cout<<"I am in B"<<endl;
12.    }
13. };
14.
15. int main(int argc, char** argv) {
16.    B b;
17.    A* aPtr=&b;
18.    aPtr->print();
19.    return 0;
20. }
21.
```

Answer: I am in A.

Question 1.4 (5 points.)

Can the following code compile? If no, give all the line numbers in the main function that the corresponding statements cannot compile.

```
1.  class A{
2.  public:
3.      int publicDataInA;
4.  protected:
5.      int protectedDataInA;
6.  private:
7.      int privateDataInA;
8.  public:
9.      A(int a,int b,int c):publicDataInA(a),protectedDataInA(b),privateDataInA(c){
10.     }
11. };
12.
13. class B: protected A{
14. public:
15.     B(int a,int b,int c):A(a,b,c){
16.     }
17.
18.     void print(){
19.         cout<<protectedDataInA<<endl;
20.     }
21. };
22.
23. int main(int argc, char** argv) {
24.     B b(10,20,30);
25.     cout<<b.publicDataInA<<endl;
26.     cout<<b.protectedDataInA<<endl;
27.     return 0;
28. }
```

Answer: Lines 25, 26.

Question 1.5 (5 points.)

List the output produced by the following program?

```
1.  class A{
2.  public:
3.      A(int x){
4.          cout<< "In A Constructor" << endl;
5.          print();
6.      };
7.      ~A() {
8.          cout<< "In A Destructor" << endl;
9.          delete _val;
10.     };
11.
12.     virtual void print() { cout<< "A."<< endl; };
13. private:
14.     char* _val;
15. };
16.
17. class B: public A{
18. public:
19.     B(int x, int y): A(x) {
20.         _dVal = new char[y];
21.         cout<< "In B Constructor 1" << endl;
22.         print();
23.     };
24.     B(): A(0) {
25.         _dVal = new char[1];
26.         cout<< "In B Constructor 2" << endl;
27.         print();
28.     };
29.     ~B() {
30.         cout<< "In B Destructor" << endl;
31.         delete _dVal;
32.     };
33.     void print() { cout<< "B" << endl; };
34. private:
35.     char* _dVal;
36. };
37.
38. int main(int argc, char** argv) {
39.     A* p1 = new B();
40.     p1->print();
41.     delete p1;
42.     return 0;
43. }
44.
```

Answer:

In A Constructor

A.

In B Constructor 2

B

B

In A Destructor

Question 1.6 (5 points.)

Will the following program be able to compile? If yes, what's the output? If no, state your reason.

```
1.  class A{
2.  public:
3.      virtual void vfa() { cout<< "In A."<< endl; };
4.  private:
5.  };
6.
7.  class B{
8.  public:
9.      virtual void vfb() { cout<< "In B." << endl; };
10. private:
11. };
12.
13.
14. class C:public A,public B{
15.     virtual void vfa() { cout<< "In C a."<< endl; };
16.     virtual void vfb() { cout<< "In C b." << endl; };
17. };
18.
19. int main(int argc, char** argv) {
20.     A* bp2 = new C;
21.     bp2->vfa();
22.     return 0;
23. }
```

Answer: In C a.

Question 1.7 (5 points.)

Consider the following main function for the program in 1.6. Replace the main function in 1.6 with the following main function. Will the program with this main function be able to compile? If yes, what's the output? If no, state your reason.

```
1.  int main(int argc, char** argv) {
2.      A* bp2 = new C;
3.      bp2->vfb();
4.      return 0;
5.  }
```

Answer: Compilation error because vfb() is not defined in class A.

Question 2 (50 points)

Longest common substring is the longest string common to a set of strings. For example, two strings “studying hard” and “playing” have the longest string “ying” in common. The following code implements the algorithm to count the length of longest common substring. It takes as input two character arrays and returns the length of the longest common subsequence of characters. The output of the following program will be:

```
Length of Longest Common Substring is 4.
```

```
1. int max(int a, int b){
2.     return (a > b)? a : b; //This statement returns true if a>b otherwise it returns false.
3. }
4. /* Returns length of longest common substring of X[0..m-1] and Y[0..n-1] */
5. int LCSubStr(char *X, char *Y, int m, int n){
6.     int count[m + 1][n + 1];
7.     int result = 0;
8.     for (int i = 0; i <= m; i++){
9.         for (int j=0; j<=n; j++){
10.            if (i == 0 || j == 0)
11.                count[i][j] = 0;
12.            else if (X[i-1] == Y[j-1]){
13.                count[i][j] = count[i-1][j-1] + 1;
14.                result = max(result, count[i][j]);
15.            }
16.            else count[i][j] = 0;
17.        }
18.    }
19.    return result;
20.}
21.void mainLCS(){
22.    char X[] = "aaaaabbbb";
23.    char Y[] = "bbbb";
24.    int m = strlen(X);  //m=9
25.    int n = strlen(Y);  //n=4
26.    cout << "Length of Longest Common Substring is " << LCSubStr(X, Y, m, n);
27.}
```

Question 2.1 The problem of the above problem is that data and algorithms are tightly coupled – the above algorithm only works with two strings. You need to refactor the above code using the Abstract Factory Pattern. In your design, the algorithms and the data are decoupled. The algorithm should be able to work with any data that conforms to a certain interface. The classes that you design need to work with the main function given below. This main function outputs the length of the longest common subsequence of integers between two sequences of integers [15, 40, 20, 30, 25] and [40, 20, 30, 1, 6], which is 3. The main function also outputs the length of the longest common substring between two strings “studying hard” and “playing”.

```
1. int main(int argc, char** argv) {
2.     vector<char> seta={'s','t','u','d','y','i','n','g',' ','h','a','r','d'};
3.     vector<char> setb={'p','l','a','y','i','n','g'};
4.     vector<int> set1={15,40,20,30,25};
5.     vector<int> set2={40,20,30,1,6};
6.     CalculateLongestSubstring calclongestsub;
7.     CompareStr* cstr=new CompareStr(seta,setb);
8.     cout<<"Length of common characters: "<<calclongestsub.LCSubStr(cstr)<<endl;
9.     CompareInt* cint=new CompareInt(set1,set2);
10.    cout<<"Length of common integers: "<<calclongestsub.LCSubStr(cint)<<endl;
11.    return 0;
12.}
```

Answer:

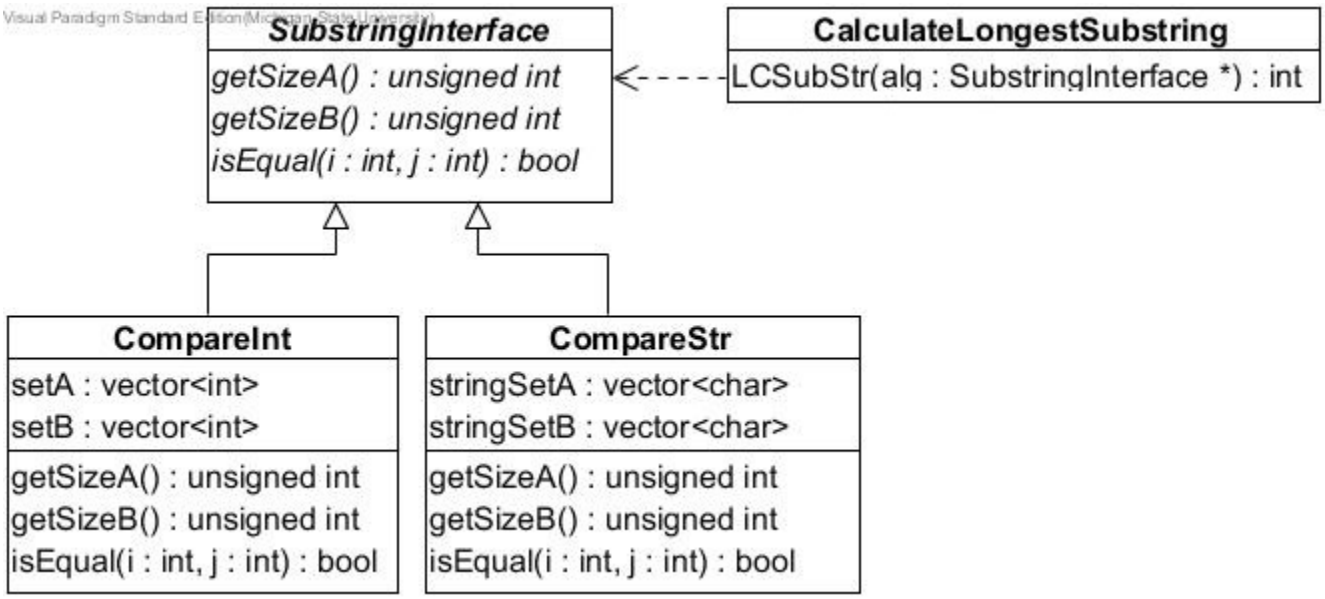
```
class SubstringInterface {
public:
    virtual unsigned int getsizeA()=0;
    virtual unsigned int getsizeB()=0;
    virtual bool isEqual(int, int)=0;
};
```

```
#include "SubstringInterface.h"
class CalculateLongestSubstring {
public:
      CalculateLongestSubstring(){}
      CalculateLongestSubstring(const CalculateLongestSubstring& orig){}
      virtual ~CalculateLongestSubstring(){}
      int max(int a, int b)
      {
           return (a > b)? a : b;
 }
int LCSubStr(SubstringInterface* alg)
{
     unsigned int m=alg->getsizeA();
     unsigned int n=alg->getsizeB();
     int LCSuff[m + 1][n + 1];
     int result = 0;
     for (int i = 0; i <= m; i++)
     {
          for (int j=0; j<=n; j++)
          {
               if (i == 0 || j == 0)
                    LCSuff[i][j] = 0;
               else if (alg->isEqual(i-1,j-1))
               {
                    LCSuff[i][j] = LCSuff[i-1][j-1] + 1;
                    result = max(result, LCSuff[i][j]);
               }
               else LCSuff[i][j] = 0;
          }
     }
     return result;
}
private:
};
```

```cpp
#include "SubstringInterface.h"
class CompareInt:public SubstringInterface{
protected:
    vector<int> setA;
    vector<int> setB;
public:
    CompareInt(vector<int>& A,vector<int>& B):setA(A),setB(B){}
    CompareInt(const CompareInt& orig){}
    virtual ~CompareInt(){}
    virtual unsigned int getsizeA(){
        return setA.size();
    }
    virtual unsigned int getsizeB(){
        return setB.size();
    }
    virtual bool isEqual(int i, int j){
        if(setA[i]==setB[j]){
            return true;
        }
        else
            return false;
    }
private:
};
```

```cpp
#include "SubstringInterface.h"
class CompareStr :public SubstringInterface{
protected:
    vector<char> stringsetA;
    vector<char> stringsetB;
public:
    CompareStr(vector<char>& A,vector<char>& B):stringsetA(A), stringsetB(B){}
    CompareStr(const CompareStr& orig){}
    virtual ~CompareStr(){}
virtual unsigned int getsizeA(){
        return stringsetA.size();
}
virtual unsigned int getsizeB(){
        return stringsetB.size();
}
virtual bool isEqual(int i, int j){
     if(stringsetA[i]==stringsetB[j]){
          return true;
      }
     else
          return false;
}
private:
};
```

Question 2.2 You need to draw the UML diagram for all your classes. Use /abc/ to denote *abc* in italic.



Question 3 (15 points)

Data normalization is a process used in statistics that takes as input a list of numbers and creates a new list of numbers with a different range of values. For example consider a list of numbers X=[100,50,50,10,10,100,20,1]. The range of values for this list is from 1 to 100. This list can be normalized by dividing each number by the maximum value i.e., 100. Thus we get X_new=[1,0.5,0.5,0.1,0.1,1,0.2,0.01] as the output. The range of values for the new list is from 0.01 to 1. Normalization can be done in other ways as well. Here are two methods to normalize a list of numbers.

1- (X-mean(X))/std(X)

2- (X-min(X))/(max(X)-min(X))

Note that for each method you need to compute two different statistics.

Method 1:

Statistics 1: mean(X) retuns the average of the list of numbers.

Statistics 2: std(X) returns the standard deviation of the list of numbers.

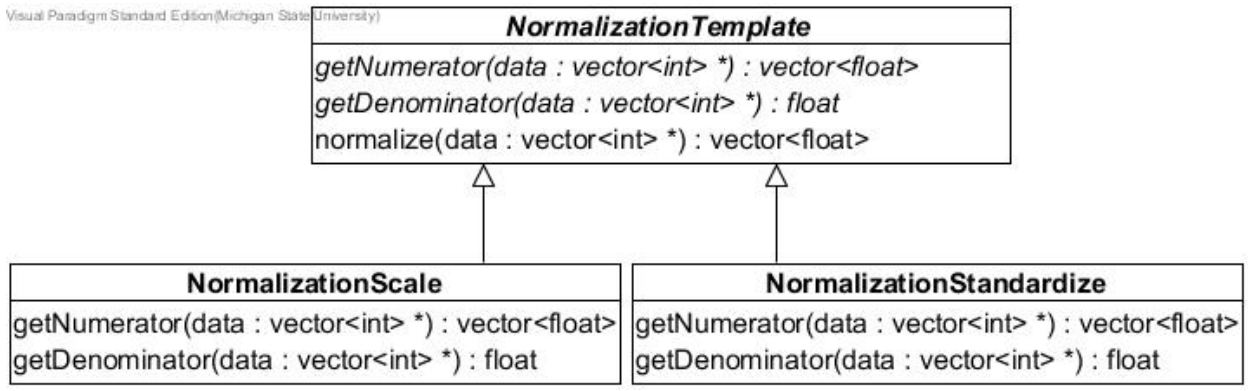
Method 2:

Statistics 1: min(X) returns the minimum value in the list of numbers.

Statistics 2: man(X) returns the maximum value in the list of numbers.

Question 3.1 (15 points) You need to draw the UML diagram to normalize the list of number using the two methods above. You must use the template method pattern. Assume X as a vector of float type values. Assume that the functions mean(X), min(X), max(X) and std(X) are given to you already. Use /abc/ to denote *abc* in italic.

Answer:



Question 3.2 (Optional. Bonus points: 20) You can write C++ code for all your classes. You do not need to write a main function.

Answer:

```cpp
class NormalizationTemplate {
public:
    virtual vector<float> getNumerator(vector<int>*)=0;
    virtual float getDenominator(vector<int>*)=0;
    vector<float> normalize(vector<int>* data){
        vector<float> normalized_data;
        vector<float> num=getNumerator(data);
        float den=getDenominator(data);
        for(int i=0;i<num.size();i++){
            normalized_data.push_back(num[i]/den);
        }
        return normalized_data;
    }
private:
};
```

```cpp
#include "NormalizationTemplate.h"
#include <cmath>
class NormalizationStandardize :public NormalizationTemplate{
public:
    NormalizationStandardize(){}
    NormalizationStandardize(const NormalizationStandardize& orig){}
    virtual ~NormalizationStandardize(){}
    virtual vector<float> getNumerator(vector<int>* data){
        vector<float> numerator;
        for(int i=0;i<data->size();i++){
            numerator.push_back(data->at(i)-datamean(data));
        }
        return numerator;
    }
    virtual float getDenominator(vector<int>* data){
        vector<float> shifted;
        shifted=getNumerator(data);
        float sumsq=0;
        for(int i=0;i<shifted.size();i++){
            sumsq=sumsq+shifted[i]*shifted[i];
        }
        return sqrtf(sumsq/shifted.size());
    }
    float datamean(vector<int>* data){
        float sum;
        sum=0;
        for(int i=0;i<data->size();i++){
            sum+=data->at(i);
        }
        return sum/data->size();
    }
private:
};
```

```cpp
#include "NormalizationTemplate.h"
#include <algorithm>
class NormalizationScale :public NormalizationTemplate{
```

```
4.  public:
5.      NormalizationScale(){}
6.      NormalizationScale(const NormalizationScale& orig){}
7.      virtual ~NormalizationScale(){}
8.      virtual vector<float> getNumerator(vector<int>* data){
9.          vector<float> numerator;
10.         auto m=*min_element(data->begin(), data->end());
11.         for(int i=0;i<data->size();i++){
12.             numerator.push_back(data->at(i)-m);
13.         }
14.         return numerator;
15.     }
16.     virtual float getDenominator(vector<int>* data){
17.         float denominator;
18.         denominator= *max_element (data->begin(), data->end())-*min_element (data-
    >begin(), data->end());
19.         return denominator;
20.     }
21. private:
22. };
```