Principles of Programming Language

Assignment 6

Student Details

Name: Krunal Rank

Adm No: U18CO081

1

```cpp
#include <bits/stdc++.h>

using namespace std;

class author
{
private:
   string author_name;

public:
   author(string a)
   {
       this->author_name = a;
   }
   string get_author_name()
   {
       return this->author_name;
   }
};

class book_publication : public author
{
private:
   string title;

public:
   book_publication(string a, string t) : author(a)
   {
       this->title = t;
   }
   string get_title()
   {
       return this->title;
```

```
    }
};

class paper_publication : public author
{
private:
    string title;

public:
    paper_publication(string a, string t) : author(a)
    {
        this->title = a;
    }

    string get_title()
    {
        return this->title;
    }
};

void display_error_message()
{
    cout << "Invalid Arguments. Arguments must follow following standards:" << endl;
    cout << "Enter 3 arguments for each record value." << endl;
    cout << "First argument should be either P or B which denotes Paper or Book
respectively." << endl;
    cout << "Second argument must be Author Name." << endl;
    cout << "Third argument must be Title." << endl;
}

int main(int argc, char **argv)
{
    if (argc % 3 != 1)
    {
        display_error_message();
        return 0;
    }
    vector<paper_publication *> p;
    vector<book_publication *> b;
    int cnt = (argc - 1) / 3;
    for (int i = 0; i < cnt; i++)
    {
        string type = string(argv[i * 3 + 1]);
        string author_name = string(argv[i * 3 + 2]);
        string title = string(argv[i * 3 + 3]);
```

```
        if (type == "P")
        {
            p.push_back(new paper_publication(author_name, title));
        }
        else if (type == "B")
        {
            b.push_back(new book_publication(author_name, title));
        }
        else
        {
            display_error_message();
            return 0;
        }
    }

    cout<<"Enter Author Name : ";
    string author_name;
    cin >> author_name;
    for(paper_publication* paper: p){
        author* a = paper;
        if(a->get_author_name()==author_name){
            cout<<"Paper Found : "<<paper->get_title()<<endl;
        }
    }
    for(book_publication* book: b){
        author* a = book;
        if(a->get_author_name()==author_name){
            cout<<"Book Found : "<<book->get_title()<<endl;
        }
    }
}
```

```
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/PPL/Assignment_6$ g++ 1.cpp
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/PPL/Assignment_6$ ./a.out B A1 B1 B A2 B2 P A1 P1
Enter Author Name : A1
Paper Found : A1
Book Found : B1
```

2

```
class Rectangle{
    private:
        double width,length;
    public:
```

```
        Rectangle(){
            this->width = 0;
            this->length = 0;
        }
        Rectangle(double x){
            this->width = x;
            this->length = x;
        }
        Rectangle(double length, double width){
            this->width = width;
            this->length = length;
        }
        double getArea(){
            return this->width*this->length;
        }
        double getPerimeter(){
            return 2.0*(this->width + this->length);
        }
};
```

3

```
#include <bits/stdc++.h>

using namespace std;

class Student
{
private:
   int id;
   vector<double> grades_obtained;
   double spi;

public:
   Student(int id)
   {
       this->id = id;
       grades_obtained = {0, 0, 0, 0, 0, 0};
       spi = 0;
   }
   Student(int id, vector<double> g_o)
   {
       if (g_o.size() != 6)
```

```
        {
            throw "Invalid Grades Obtained Size! It should be 6!";
        }
        for (auto i : g_o)
            if (i < 0 || i > 5)
                throw "Invalid Grade! Grade should be between 0.0 and 5.0 inclusive.";
        this->id = id;
        grades_obtained = g_o;
        this->calculate_spi();
    }

    void calculate_spi()
    {
        this->spi = accumulate(this->grades_obtained.begin(),
this->grades_obtained.end(), 0.0) / 6.0;
    }

    void display()
    {
        cout << "Student ID : " << this->id << endl;
        cout << "Scores in Subjects : ";
        for (auto i : this->grades_obtained)
            cout << i << " ";
        cout << endl;
        cout << "SPI : " << this->spi << endl;
        cout << endl;
    }
};

int main()
{
    int n = 6;
    vector<Student*> s;
    for (int i = 0; i < n; i++)
    {
        vector<double> g;
        for (int j = 0; j < 6; j++)
            g.push_back((double)(rand() % 5));
        s.push_back(new Student(i, g));
    }
    for (auto i : s)
        i->display();
}
```

```
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/PPL/Assignment_6$ ./a.out
Student ID : 0
Scores in Subjects : 3 1 2 0 3 0
SPI : 1.5

Student ID : 1
Scores in Subjects : 1 2 4 1 2 2
SPI : 2

Student ID : 2
Scores in Subjects : 0 4 3 1 0 1
SPI : 1.5

Student ID : 3
Scores in Subjects : 2 1 1 3 2 4
SPI : 2.16667

Student ID : 4
Scores in Subjects : 2 0 2 3 2 0
SPI : 1.5

Student ID : 5
Scores in Subjects : 4 2 2 3 4 2
SPI : 2.83333
```

4.

```cpp
#include <bits/stdc++.h>

using namespace std;

class ResourceStatus
{
private:
   vector<vector<int>> statusRef;

public:
   ResourceStatus()
   {
       statusRef = vector<vector<int>>(3, vector<int>(3, 0));
   }
   ResourceStatus(vector<vector<int>> s)
   {
       statusRef = s;
       this->processStatusCount();
   }

   void processStatusCount()
   {
       try
       {
           vector<int> cnt(3, 0);
```

```
            for (auto i : statusRef)
                for (auto j : i)
                    cnt[j]++;
            cout << "Process Status Count: " << endl;
            cout << "No. of Free Resources : " << cnt[0] << endl;
            cout << "No. of Occupied Resources : " << cnt[1] << endl;
            cout << "No. of Inaccessible Resources : " << cnt[2] << endl;
            if (cnt[1] > cnt[0])
                throw -1;
        }
        catch (int x)
        {
            if (x == -1)
            {
                cout << "Exception : Occupied Resources exceed available free
resources!" << endl;
                for (int i = 0; i < 3; i++)
                    for (int j = 0; j < 3; j++)
                        if (this->statusRef[i][j] == 2)
                            this->statusRef[i][j] = 0;
            }
        }
    }
};

int main(int argc, char **argv)
{
    if (argc != 10)
    {
        cout << "Invalid Arguments. 9 arguments are required. Each argument must have
value either 0,1,2 based on resource status." << endl;
        return 0;
    }
    vector<vector<int>> status(3, vector<int>(3, 0));
    for (int i = 1; i < 10; i++)
    {
        int row = (i - 1) / 3;
        int col = (i - 1) % 3;
        int val = atoi(argv[i]);
        if(val<0 || val>2){
            cout<<"Invalid Arguments. Each argument must have value either 0,1,2 based
on resource status."<<endl;
            return 0;
        }
        status[row][col] = val;
```

```
    }
    ResourceStatus *rs = new ResourceStatus(status);
}
```

```
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/PPL/Assignment_6$ ./a.out 0 0 0 1 1 1 1 2 24
Invalid Arguments. Each argument must have value either 0,1,2 based on resource status.
kr@arc-warden:/mnt/6AD574E142A88B4D/BTech/Assignments/4th_Year/PPL/Assignment_6$ ./a.out 0 0 0 1 1 1 1 2 2
Process Status Count:
No. of Free Resources : 3
No. of Occupied Resources : 4
No. of Inaccessible Resources : 2
Exception : Occupied Resources exceed available free resources!
```