**Assignment – 3**

Subject: COP- 3003 Programming -II

Instructor: Dr. Deepa Devasenapathy

Total points:100

1.Write a program to access members of a STUDENT class using pointer to object members (10 points)

=========================SourceCode================================

#include <iostream>  
using namespace std;  
class Student {  
public:  
 int UIN;  
 string name;  
 Student(int UIN, string name) {  
 this->UIN = UIN;  
 this->name = name;  
 }  
};  
int main() {  
 //Creating objects of class 'Student'.  
 Student x0(10001, "Alex");  
 Student x1(12345, "Alexander");  
  
 //Points to objects  
 Student \*ptr[2];  
  
 //points to the address of the object in order to access/assign it.  
 ptr[0] = &x0;  
 ptr[1] = &x1;  
 //For displaying student info.  
 //Loop to cout each students' information(in this case, 2).  
 for (int num\_of\_studs = 0; num\_of\_studs < 2; num\_of\_studs++) {  
 cout << string(20, '-') << endl;  
 //'ptr[]' points to the value of said variable in order to access it.  
 cout << "UIN : " << ptr[num\_of\_studs]->UIN << endl;  
 cout << "Name : " << ptr[num\_of\_studs]->name << endl;  
 }  
 cout << string(20, '-') << endl;//aesthetics  
 return 0;  
}

=========================Output====================================

--------------------

UIN : 10001

Name : Alex

--------------------

UIN : 12345

Name : Alexander

--------------------

Process finished with exit code 0

===================================================================

2.Write a Program to generate Fibonacci Series by using Constructor to initialize the Data Members. (10 points)

=========================SourceCode================================

//Formula -> [x\_n = x\_n−1 + x\_n−2]  
#include <iostream>  
using namespace std;  
class Fibonacci\_Series {  
private:  
 int x0, x1, x\_sum, compute\_limit;  
public:  
 Fibonacci\_Series(int x0, int x1, int compute\_limit){  
 //will cause an infinite loop if values are both '0'->  
 //->or if values are not set at all, there will be a logical error.  
 this->x0 = x0;  
 this->x1 = x1;  
 //using pointers to make accessible in 'compute\_fib\_series' function  
 this->compute\_limit = compute\_limit;  
 }  
 void compute\_fib\_series(){  
 cout << "{";  
 while(x\_sum <= compute\_limit){  
 x\_sum = x0 + x1;  
 cout << " <" << x\_sum << ">, ";  
 x0 = x1;  
 x1 = x\_sum;  
 x\_sum = x0 + x1;  
 }cout << "\b\b } ";//'\b\b' overrides the trailing comma.   
 }  
};  
  
int main(){  
 //format: <x0, x1, compute\_limit(max value)>  
 Fibonacci\_Series fibo(6, 8, 1200);  
 fibo.compute\_fib\_series();  
 return 0;  
}

}

=========================Output====================================

{ <14>, <22>, <36>, <58>, <94>, <152>, <246>, <398>, <644>, <1042> }

Process finished with exit code 0

===================================================================

3. Write a Program illustrating how the constructors are implemented and the order in which they are called when the classes are inherited. Use three classes named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha & beta. (10 points)

=========================SourceCode================================

#include <iostream>  
#include <string>  
using namespace std;  
//global variable  
string id;  
class alpha{  
public:  
 alpha(){  
 id = "Base-alpha";  
 cout << id << endl;  
 }  
};  
class beta{  
public:  
 beta(){  
 id = "Base-beta";  
 cout << id << endl;  
 }  
};  
//By switching the order when defining derived class, ->  
// ->you switch the order at which they are called during runtime.  
class gamma : public beta, public alpha{  
public:  
 gamma(){  
 id = "Derived-gamma";  
 cout << id << endl;  
 }  
 //Code below will switch the order base classes are called during runtime.  
 //Output flow would then be: alpha->beta->gamma.  
/\*  
class gamma : public alpha, public beta{  
public:  
 gamma(){  
 id = "Derived-gamma";  
 cout << id << endl;  
 }  
\*/  
};  
int main(){  
 //Calls class 'beta' first then class 'alpha'(bottom->up) ->  
 // ->then, since gamma is derived, it is called last.  
 gamma();  
 return 0;  
}

=========================Output====================================

Base-beta

Base-alpha

Derived-gamma

Process finished with exit code 0

===================================================================

4. Write a program showing implementation of stack class having the functionality of

1. Push
2. Displaying stack elements
3. Size of the stack
4. Display top of the stack
5. Pop
6. Checking whether stack is empty. (10 points)

=========================SourceCode================================

#include <iostream>

#include <stack>

using namespace std;

//stores stack to be able to edit it

class stackoperations{

public:

stack<string> stack1;

string stack\_elem;

};

//derived class for UI funtions

//not helpful, just throwing it in

class ui\_elem: public stackoperations{

public:

//Adds enough blank lines to make it seem as if the console was cleared.

//Every other built-in function I know of or could find was OS-dependant.

void clear\_console(){

cout << string(10, '\n') << endl;

}

//visually separates the output from the following menu prompt.

void in\_out\_sep(){

cout << string(100, '#') << "\n" << endl;

}

//Wraps user's menu selection with brackets.

void input\_box(){

cout << "[ ]\b\b";

}

};

//to access objects within class

stackoperations s{};

ui\_elem ui{};

//(a)Display the elements in stack

void displ\_menu(){

ui.in\_out\_sep();

cout << "Enter\n[1]Add Element\n[2]Remove Previous Element\n[3]Display Quantity of Elements\n[4]Call previously added Element\n"

"[5]Display All Elements\n[0]Exit" << endl;

}

//(1/2(d))Pop some stack values.

void add\_elem(){

cin >> s.stack\_elem;

s.stack1.push(s.stack\_elem);

ui.clear\_console();

}

void rem\_elem(){

s.stack1.pop();

ui.clear\_console();

}

//(b)get the size of stack

void displ\_stack\_size(){

//clearing before so that user can see ouput

//could use a continue-verification or timer

ui.clear\_console();

cout << "Quantity: " << s.stack1.size() << "\n" << endl;

}

//(c)Print top of the stack

void displ\_last\_elem(){

ui.clear\_console();

cout << s.stack1.top() << "\n" << endl;

}

//(a)&(1/2(d))Display the elements in stack

void displ\_all\_elem(){

ui.clear\_console();

//creating a copy of stack to use in function so that the original stack does not get modified.

stack<string> copy\_stack1 = s.stack1;

while(!copy\_stack1.empty()){

cout << copy\_stack1.top() << "\n";

copy\_stack1.pop();

}}

int main(){

int usr\_choice;

do{

//Outputs menu options

displ\_menu();

ui.input\_box();

cin >> usr\_choice;

switch(usr\_choice){

case 0:

break;

case 1:

add\_elem();

break;

case 2:

rem\_elem();

break;

case 3:

displ\_stack\_size();

break;

case 4:

displ\_last\_elem();

break;

case 5:

displ\_all\_elem();

break;

}

}while(usr\_choice != 0);

return 0;

}

=========================Output====================================

//Not exact output, reduced spacing/empty lines for submission

####################################################################################################

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[1]

First\_Element

####################################################################################################

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[3]

Quantity: 1

####################################################################################################

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[4]

First\_Element

####################################################################################################

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[5]

First\_Element

####################################################################################################

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[0]

Process finished with exit code 0

===================================================================

5. Write a Program to design a class having static member function named showcount() which has the property of displaying the number of objects created of the class. (10 points)

=========================SourceCode================================

#include <iostream>  
using namespace std;

class counter{  
public:  
 int obj\_value;  
 //static variable for static function  
 static int cnt;  
 //constructor  
 counter(){  
 int objs = ++cnt;  
 obj\_value = 0;  
 }  
 //deconstructor  
 ~counter(){--cnt;}  
 //loop to create obj to give something to count  
 static void obj\_loop(){  
 int inp;  
 cnt = 0;  
 do{  
 //create new obj  
 counter obj;  
 cout << "Enter an int to for obj: " << endl;  
 cin >> obj.obj\_value;  
 ++cnt;//increment/keeps track of count for each iteration  
 cout << "Enter '0' or non-int value to exit" << endl;  
 cin >> inp;//check if user wants to exit  
 }while(inp != NULL);//continues loop as long as user inputs any int except '0'  
 }  
 /\*Static function purpose:  
 >can be called when an object from a class hasn't been initialized.  
 >can only access other static variables of same class.\*/  
 static void show\_count(){cout << "Count [" << cnt << "]" << endl;}//displays count  
};  
int counter::cnt;  
int main() {  
 counter::obj\_loop();  
 counter::show\_count();  
 return 0;  
}

=========================Output====================================

Enter an int to for obj:

1

Enter '0' or non-int value to exit

2

Enter an int to for obj:

1

Enter '0' or non-int value to exit

0

Count [2]

Process finished with exit code 0

===================================================================

6. Using multilevel inheritance write a program to manage patient bills in a hospital. (20 points)

=========================SourceCode================================

//Using multilevel inheritance write a program to manage patient bills in a hospital.

#include <iostream>

#include <string>

using namespace std;

class patient {

//members within protected are accessible to derived classes, not to outside of class

protected:

int patient\_id;

string name;

public:

int cover\_charge = 250;

patient(int pat\_id, string pat\_name) {

patient\_id = pat\_id;

name = pat\_name;

}

void pat\_inf() {

cout << "Patient ID: " << patient\_id << endl;

cout << "Patient Name: " << name << endl;

}

};

class newpatient : public patient {

protected:

int days;

double daily\_rate;

public:

//constructor for newpatient class

newpatient(int patient\_id, string name, int d, double dr) : patient(patient\_id, name) {

days = d;

daily\_rate = dr;

}

};

//derived class from child class newpatient. Parent class is patient.

class payment: public newpatient {

public:

double amount\_paid;

double subtotal = days \* daily\_rate + cover\_charge;

//constructor

payment(int patient\_id, string name, int d, double dr) : newpatient(patient\_id, name, d, dr) {}

//function to calculate call to pay bill and update amount due

double pay(int amt){

subtotal = subtotal - amt;

return subtotal;

}

void bill\_inf(){

//patient name and id are protected

patient::pat\_inf();

cout << "Number of days in hospital: " << days << endl;

cout << "Daily rate: " << daily\_rate << endl;

cout << "Total bill: " << subtotal << endl;

}

double bill\_pay() {

cout << "Amount due: " << subtotal << "\nEnter Amount Paid: " << endl;

cin >> amount\_paid;

pay(amount\_paid);

cout << "Amount due: " << subtotal << endl;

}

};

int main() {

newpatient p1(1001, "John Smith", 5, 250.00);

p1.pat\_inf();

payment p2(1001, "John Smith", 5, 250.00);

p2.bill\_inf();

p2.bill\_pay();

return 0;

}

=========================Output====================================

Patient ID: 1001

Patient Name: John Smith

Patient ID: 1001

Patient Name: John Smith

Number of days in hospital: 5

Daily rate: 250

Total bill: 1500

Amount due: 1500

Enter Amount Paid:

700

Amount due: 800

Process finished with exit code 0

===================================================================

7.Write a program to find the shapes using Function overloading. (10 points)

=========================SourceCode================================

#include <iostream>

using namespace std;

class Shape{

public:

double area{0};

string shape{""};

//Only one measurement value is needed to find the area of a square.

//If user only enters one measurement, then the shape will be overloaded to 'Square'.

float get\_area(float l){

float length = l;

area = (length \* length);

shape = "Square";

return area;

}

//overloading the 'area' function.(Able to change implementation depending on given parameters)

float get\_area(float l, float w) {

float length = l;

float width = w;

area = (length \* width) / 2;

shape = "Triangle";

return area;

}

};

//If the functions were to be in seperate, derived classes, then this would be Overriding.

int main(){

Shape s{};

//overloads 'get\_area' function to calculate for square since there is only one value given

s.get\_area(5.00);

cout << "Shape: " << s.shape << "\n\tarea: " << s.area << endl;

//overloads 'get\_area' function to calculate for triangle since there are 2 values given

s.get\_area(5.00, 8.00);

cout << "\nShape: " << s.shape << "\n\tarea: " << s.area << endl;

return 0;

}

=========================Output====================================

Shape: Square

area: 25

Shape: Triangle

area: 20

Process finished with exit code 0

====================================================================

8. Write a program to maintain employee database using virtual class.

Hint:

1. Create employee class details such as name, id
2. Create account class that has the pay, bonus etc.,
3. Create admin class that had experience, designation etc.,
4. create a master class that switches to
5. Enter the details of the employee
6. Display the details of the employee
7. Update information if needed
8. Quit the program. (20 points)

=========================SourceCode================================

=========================Output====================================

===================================================================