In Class Assignment Week-9

1. Show two type of Inheritance program of your own choice.  
  
2. Using multilevel inheritance write a program to manage patient bills in a hospital.  
  
3. Use friend function to find the maximum of two numbers from different classes.

**[Question 1.1]**

# #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;  
}

**[Question 1.1-Output]**

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

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

Element1

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

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]

Elem2

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

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

[0]Exit

[2]

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

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]

Element1

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

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]

Element1

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

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

Enter

[1]Add Element

[2]Remove Previous Element

[3]Display Quantity of Elements

[4]Call previously added Element

[5]Display All Elements

Element1

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

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

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

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]

Element1

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

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]

Element1

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

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

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]

Element1

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

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]

Element1

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

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

Process finished with exit code 0

**[Question 1.2]**

#include <iostream>  
using namespace std;  
  
class Shapes{  
public:  
 int area;  
};  
Shapes s{};  
class triangle : public Shapes{  
public:  
 int area(int l, int w){  
 int area\_triangle = (l \* w) / 2;  
 return s.area = area\_triangle;  
 }  
};  
class rectangle : public Shapes{  
public:  
 int area(int l, int w){  
 int area\_rectangle = (l \* w);  
 return s.area = area\_rectangle;  
 }  
};  
  
int main() {  
 int l, w;  
 triangle tri{};  
 rectangle rect{};  
 cout << "Enter Length and Width: " << endl;  
 cin >> l >> w;  
 tri.area(l, w);  
 cout << "Area if Triangle: " << s.area << endl;  
 rect.area(l, w);  
 cout << "Area if Rectangle: " << s.area << endl;  
 return 0;  
}

**[Question 1.2-Output]**

Enter Length and Width:

45

5

Area if Triangle: 112

Area if Rectangle: 225

Process finished with exit code 0

**[Question 2]**

//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;  
}

**[Question 2-Output]**

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:

500

Amount due: 1000

Process finished with exit code 0

**[Question 3]**

#include<iostream>  
using namespace std;  
class y\_nums;  
class x\_nums;  
  
class x\_nums{  
private:  
 int x;  
public:  
 x\_nums(int x)  
 {  
 this->x = x;  
 }  
 //can access private members of y\_nums & x\_nums  
 friend void compare(x\_nums,y\_nums);  
};  
class y\_nums{  
private:  
 int y;  
public:  
 y\_nums(int y)  
 {  
 this->y=y;  
 }  
 friend void compare(x\_nums,y\_nums);  
};  
void compare(x\_nums a,y\_nums b)  
{  
 if(a.x > b.y)  
 {  
 cout<< "Maximum: x = " << a.x <<endl;  
 }  
 else if (a.x < b.y)  
 {  
 cout<< "Maximum: y = " << b.y <<endl;  
 }  
 else  
 {  
 cout<< "Both are equal." << endl;  
 }  
}  
  
int main()  
{  
 x\_nums x(10);  
 y\_nums y(9);  
 compare(x,y);  
 return 0;  
}

**[Question 3-Output]**

Maximum: x = 10

Process finished with exit code 0

##\*If x = 10 and y = 10\*##

\*\*-------------------[Output]-----------------------\*\*

Both are equal.

Process finished with exit code 0

**[Question 4]**

#include <iostream>  
using namespace std;  
class Shapes{  
public:  
 int area;  
};  
Shapes s{};  
class triangle : public Shapes{  
};  
//derived class 'right\_triangle' from base class 'triangle' which is derived from base class 'Shapes'  
class right\_triangle : public triangle{  
public:  
 int area(int l, int w){  
 int area\_right\_triangle = (l \* w) / 2;  
 return s.area = area\_right\_triangle;  
 }  
};  
class rectangle : public Shapes{  
public:  
 int area(int l, int w){  
 int area\_rectangle = (l \* w);  
 return s.area = area\_rectangle;  
 }  
};  
int main() {  
 int l, w;  
 triangle tri{};  
 rectangle rect{};  
 right\_triangle rt{};  
 cout << "Enter Length and Width: " << endl;  
 cin >> l >> w;  
 rt.area(l, w);  
 cout << "Area if Right Triangle: " << s.area << endl;  
 rect.area(l, w);  
 cout << "Area if Rectangle: " << s.area << endl;  
 return 0;  
}

**[Question 4-Output]**

Enter Length and Width:

8

9

Area if Right Triangle: 36

Area if Rectangle: 72

Process finished with exit code 0