MID TERM EXAM SOLUTION - BESE15
OBJECT ORIENTED PROGRAMMING PARADIGM

Question 1:

ii, iii, iv are legal
In (i), private data member price is not accessible
In (v) and (vi), private member function getProfit() is being called which is not accessible here

Question 2:

```
const int x=17; // value of x cannot be changed throughout
//the program
class A
{
        public:
                A();
                A(int n);
                int f() const; //function f() is constant which
        //means this function will not change the value of //class
        data members
                int g(const A& x); //object x is being passed as
        //constant so x cannot be changed within this function
        private:
                int i;
};
```

Question 3:

(a)

```
class stack
{
  public:
      void push(const int data) {arr[top++] = data; }
       int pop() const {return arr[--top];} //cannot modify class
//data member top; correction: remove the keyword const
  protected:
      int arr[100];
  private:
      int top = 0; //cannot initialize a class member here;
//correction: use a public constructor or class member function //to
assign values to class data members
};
```

(b)

```
class Base{
public:
        void init() { count = sum = num =0; } //const num cannot be
//assigned a value here; correction: use constructor
//initialization list to initialize a constant data member
protected:
        int count;
private:
        int sum;
```

```
    const int num;
};

class Derived: public Base
{
       public:
       void init() { avg = 0;}
        int getSum()  { return sum;} //sum is a private data member
//of class Base which is not accessible in the derived class;
correction: make sum protected or make a public function in class
Base which returns sum and then call that public function in the
derived class function getSum()
    private:
       int avg;
    };
```

Question 4:

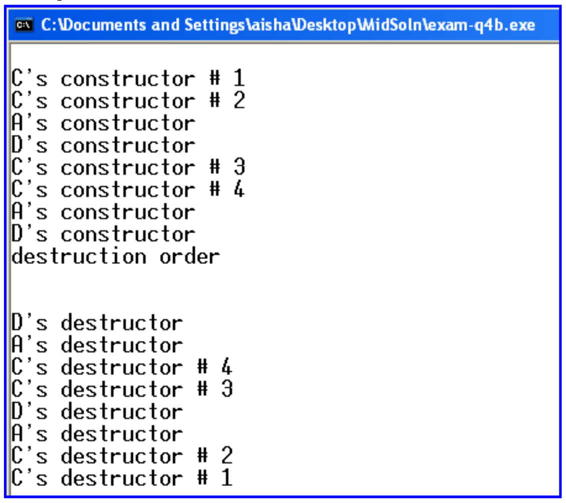
(a)

```
        class ss
        {
               static int c;
               public:
                      static void set() { c++; }

                void display(){cout<< c; }
        };
        int ss::c=12; //c = 12

        void main()
        {
               ss obj;
               obj.set(); //c = 13
               ss::set(); // c = 14
               obj.display(); // displays the number '14'
               getch();
        }
```

(b) Output is:



```
C:\Documents and Settings\aisha\Desktop\MidSoln\exam-q4b.exe

C's constructor # 1
C's constructor # 2
A's constructor
D's constructor
C's constructor # 3
C's constructor # 4
A's constructor
D's constructor
destruction order


D's destructor
A's destructor
C's destructor # 4
C's destructor # 3
D's destructor
A's destructor
C's destructor # 2
C's destructor # 1
```

Question 5:

```
void main()
{
 A a;
 B b;
 C c;
 E e;

//a.i2=1;          //protected member i2 of class A is not accessible;
correction: make a public function seti2 which assigns the argument passed
to this function to the data member i2, and then call a.seti2(1)

 b.A::seti(3);
 a.seti(2);

 //b.f1("INPUT1");   //function f1() of class B over-rides the f1() in
class A, so it requires an integer as an argument;
correction: pass an integer as an argument or call the function as
b.A::f1("input2");


//c.seti(4);          //class C is derived from class A using private
inheritance so all public functions of A have become private in C and
private functions are not accessible here;
correction: use public inheritance while deriving class C from class A

//c.f1("INPUT2");      //same as above

 e.seti(5);
 e.B::i3=7;

//e.f1("INPUT3");      //class E is derived from both B and D, and both of
these classes implement the function f1(), so there is an ambiguity
between which function should be called;
Correction: use the scope resolution operator to call the correct function
like e.B::f1("Input3");
}
```

Question 6:

```
//your solution may differ in certain places

#include<iostream>
#include<conio>
class time
{
   private:
      int h,m;
   public:

      time(int hr=0, int min=0):h(hr),m(min){}
       void display()  const
       {
            cout<<"time is:"<<h<<":"<<m;
          }
          int geth()const {return h;}
          int getm()const {return m;}
          friend time operator++(time&,int);
          friend void swap(time&,time&);
};

time operator++(time& x,int)
{
   time temp = x;
   x.m++;
   if(x.m>59)
   {
      x.m=0;
       x.h++;
   }
   return temp;
}


void swap(time& t1, time& t2)

{
   time temp;
   temp.h=t1.h;
   temp.m=t1.m;
   t1.h=t2.h;
   t1.m=t2.m;
   t2.h=temp.h;
   t2.m=temp.m;
   /* OR you can also write the following lines of code
   time temp;
   temp = t1;   //automatically calls the default assignment operator for the object
   t1 = t2;
   t2 = temp; */
}
```

```
class sectime:public time
{
     int s;
   public:
     sectime(int hr=0, int min=0, int sec=0):time(hr,min),s(sec){}
      void display() const
      {
           time::display();
         cout<<":"<<s;
      }
      sectime operator+(const sectime& sec)const
           {

        int h = sec.geth() + geth();
        int m = sec.getm() + getm();
        int second = sec.s+ s;
        return sectime(h,m,second);
      }

};

void main()
{
   time t1(10,20);
   time t2(4,50);
   time x = t1++;
   x.display();
   cout<<endl;
   cout<<"t1 ";t1++.display();cout<<endl;
   swap(t1,t2);
   cout<<"After swapping \t";
   cout<<"t1: ";t1.display(); cout<<'\t';
   cout<<"t2: ";t2.display(); cout<<endl<<endl;
   sectime s1(10,20,30);
   sectime s2(1,2);
   s1.display();cout<<endl;
   s2.display();
   sectime s3 = s1 + s2;
   cout<<"\nafter addition  ";   s3.display();
   getch();
}
```