Assignment - 2

Course : OOP - 2D

Dept : BS - CS

Name: Hazan NOOR

Section: 2D

* Question - 1 *

Source-Code:

```
#include <iostream>
using namespace std;
class company {
 public:
  static int count_employees; // Static Variable
  company(){
        count_employees++;
  }
  static void display_count(){ // Static function
        cout << "Total NO: of Employees: << count_
     employees << endl;
  }

};
  int company :: count_employees = 0;
 int main(){
   company c1;
   company c2;
   company c3;
   company:: display-count();
      return 0;
}
```

* Question - 2 *

Source - code :

```
#include <iostream>
using namespace std;
class Employee {
   int emp-id;
   const int salary;  // constant variable
public:
  Employee (int id , int sa): emp-id(id) , Salary(sa)
  {
        cout<< "Address of current object : "<< this<<endl;
        cout<<" Employee ID : " this -> emp-id <<endl;
        cout<< "Salary : " this -> salary << endl;
       cout<< " _ _ _ _ _ _ _ _ _ _ _ _ _ _ "<<endl;
  }
};
int main() {
     Employee e1 (101, 30000);
     Employee e2 (102, 35000);
     Employee e3 (103, 40000);
     return0;
}
```

* Question - 3 *

Source - Code :

```
# include <iostream>
 using namespace std;

class car {
        string model;
         int year;
  Public:        // ~~Dstt~~ Constructor
     car ( string m , int y ) : model (m), year (y) {
        cout << "Car Created -> Model : " << model <<
     " , Year:" << year << endl;
   }
~ car () {        // Destructor
     cout << "Car Destroyed -> Model: " << model <<
     ", Year:" << year << endl;
  }
};

  int main () {
    car c1 ("Honda" , 2008);
    car c2 (" Suzuki" , 2010);
    cout << "End of Main Function" << endl;
       return 0;
 }
```

* Question - 4 *

Source - Code:

```
#include <iostream>
using namespace std;

 class Book {
        string title ;
        string author;
 public:
      Book (string t , string a) {
       title = t;
        author = a;
    }               // Display function
     void display () {
      cout << " Book Title is: " << title << endl;
      cout << " Author of Book is : " << author << endl;
    }

};

  int main () {
      Book* ptr = new Book ("Alchemist", "Coulo Poelo");
       ptr -> display ();
       delebe ptr;
        ptr = nullptr;
       return 0;
  }
```

* Question - 5 *

Source code:

```
#include <iostream>
  Using namespace std;
class  rectangle {
       public:
        int length ;
        int breadth;       //copy constructor
       rectangle (const rectangle &r)
       {
             Length  = r.Length ;
             breadth  =  r.breadth;
       }
       rectangle ( int l  , int b ){
             Length = l ;
             breadth = b;
       }
};
  int main () {
       rectangle r1 (2,3);
       rectangle r2 = r1 ;
       cout<< r2.length<<endl;
       cout<< r2 .breadth << endl;
       return0;
  }
```

* Question - 6 *

Source -code:

```
# include <iostream>
# include< cstring>
~~#~~ using namespace std;
 Class person {
  Private:
       char *name;
  Public:
          // constructor
   Person (const char *n) {
        name = new char [strlen(n) + 1];
         strcpy (name, n);
    }
      // shallow copy constructor
   Person (const person & p){
          name = p.name;
       cout<< "shallow copy"<<endl;
   }
    Person & operator = (const person & p) {
            if ( this != &p)
              delete [] name;
      name = new char [strlen (p.name) + 1];
      strcpy (name, p.name);
      cout<< "Deep Assignment operator";
  }
  return *this;
  void changename (const char* new) {
             strcpy(name, new);
        }
    void display() const{
        cout << "Name: " << name << endl;
    }
 ~ person(){
        delete [] name;
    }
};
int main() {
    cout << "Original person" << endl;
    person P1 ("AliNOOR");
    P1. display();
    cout << " Creating shallow copy" << endl;
      // shallow copy.
       person P2 = P1;
       P2. display ();
       P1 . change name ("Ahmed");
         P2. display();
    // Deep copy using assignment OP
    Person P3 ("Dady");
       P3 = P1 ;
       P3. display();
       P1 . changename ("zainab");
       P1 . display ();
       P3 . display ();
    return 0;
}
```

* Question - 7 *

Source - code:

```
#include <iostream>
#include <string>
using namespace std;
  // Base class
class Employee {
   public:
       string name;
        int id;
        int salary;
     // constructor
 Employee (string n, int i, int s) {
              name = n;
               id = i;
               Salary = s;
       }
              // Display function.
 void Display () {
       cout << "Name : " << name << endl;
      cout << " ID : " << id << endl;
      cout << "salary:" << salary << endl;
      }

};    // inheritance -> Derived class.
 class Manager : Public Employee {
  Public:
           String department;
     Manager (String n, int Id , int s , string d)
            : Employee (n, Id , s) {
                  department = d;
            }
void display (){
   Employee ::display ();
   cout "department : " << department << endl;
   }

};

int main () {
      // Base class
   Employee emp("Azan",100, 30000);
   emp.display();
   cout << endl;
   // Derived class
   Manager m ("Khalid" 100, 3000 , "HR");
    m.display();
     Return 0;
 }
```

* Question - 8 *

Source - code:

```
# include <iostream>
 using namespace std;
 class Shape {   // Base class
   Private:
       double area;
   Protected:
     ~~Virtual~~ void area calculation () {
           area = 0;
     }
     // this method will allow overriding.

   void set area (double a) {
           area = a;
       }
   Public:
     void display () {
         ~~calculate area();~~ calculate area ();
     cout << "Area : " area <<endl;
     }
 };
  class circle : Public Shape {
   Private:
       double radius;
   Protected:
         // overriding
    void calculate area () override {
       double area = 3.14 * radius * radius;
           Set area (area);
      }
Public:
        //constructor
    circle (double r) {
            radius = r;
        }
};
    int main () {
        circle c (4);
       c.display ~~area~~ ();
        return 0;
    }
```

*** Question - 9 ***

Source - code:

```
 #include <iostream>
  Using namespace std;
  Class vehicle {    //Base class
  Public:
        virtual void startengine () {
            cout << " Engine started "<< endl;
        }
};              /// Derived class
class Airplane : public vehicle {
   public:
    void startengine () override {
         cout << "Airplane Engine started"<< endl;
    }

};
```

```
int main(){
    vehicle* vehi ; // Base class pointer
     Airplane  airplane ; // Derived class
      vehi = & airplane ; // Base ptr to derived.
      vehi → startengine () ;
         return 0 ;
    }
```

*** Question - 10 ***

Source code:

```
 #include <iostream>
  # include < string>
    using namespace std;
            //Base class
     class person {
    Private:
           string name;
            int age ;
   Public:
         void setdata( string n, int a) :
                name (n) , age (a) { }
         void showdata  () {
          cout << "Name : " << name << endl;
          cout << " Age : " << age << endl;
        }
 };
   class student : public person {
    private :
```

```
char grade;
public: //setter
        void setgrade (char g): grade(g){ }
        void show~~S~~studentdata () {
            showdata ();
          cout<<"Grade :" grade<<endl;
        }
};

int main () {
    Student s;
    s.setdata ("Ali", 19);
    s. set grade ('A');
    s. show student data(); // displaying
                            // data
     return 0;
  }
```