Answer1:

Code:

```cpp
#include<iostream>
#include<string>

using std::cout;
using std::cin;
using std::string;

class Traveler {
private:
    string NameOfTraveler;
public:
    Traveler() = delete;

    Traveler(string NameOfTraveler_) {
        this->NameOfTraveler = NameOfTraveler_;
    }

    Traveler(Traveler& traveler_) {
        this->NameOfTraveler = traveler_.getNameOfTraveler();
    }

    string getNameOfTraveler()const {
        return this->NameOfTraveler;
    }

    void setNameOfTraveler(string NameOfTraveler_) {
        this->NameOfTraveler = NameOfTraveler_;
    }
};

class Pager {
private:
    string NumberOfPager;
public:
    Pager() = delete;

    Pager(string NumberOfPager_) {
        this->NumberOfPager = NumberOfPager_;
    }

    Pager(Pager& pager_) {
        this->NumberOfPager = pager_.getNumberOfPager();
    }

    string getNumberOfPager()const {
        return this->NumberOfPager;
    }

```

```
49      void setNumberOfPager(string& NumberOfPager_) {
50          this->NumberOfPager = NumberOfPager_;
51      }
52  };
53
54  class BusinessTraveler :public Traveler {
55  private:
56      Pager PagerOfBusinessTraveler{ "4001234567" };
57  public:
58      BusinessTraveler() :Traveler("xiaoming") {};
59
60      BusinessTraveler(string& NameOfBusinessTraveler_) :Traveler(NameOfBusinessTraveler_) {};
61
62      BusinessTraveler(string& NameOfBusinessTraveler_, string& NumberOfPagerOfBusinessTraveler_)
63          :Traveler(NameOfBusinessTraveler_) {
64          this->PagerOfBusinessTraveler = Pager(NumberOfPagerOfBusinessTraveler_);
65      }
66
67      BusinessTraveler(BusinessTraveler& businesstraveler_)
68          :Traveler(businesstraveler_.getNameOfTraveler()) {
69          this->PagerOfBusinessTraveler = Pager(businesstraveler_.PagerOfBusinessTraveler.getNumberOfPager());
70      }
71
```

```
72      string BusinessTravelerToString()const {
73          return this->PagerOfBusinessTraveler.getNumberOfPager() + "   " + this->getNameOfTraveler();
74      }
75
76      void setPagerOfBusinessTraveler(const Pager& pager_) {
77          this->PagerOfBusinessTraveler = pager_;
78      }
79
80      BusinessTraveler& operator=(const BusinessTraveler& bt_){
81          this->setNameOfTraveler(bt_.getNameOfTraveler());
82          this->PagerOfBusinessTraveler = Pager(bt_.PagerOfBusinessTraveler.getNumberOfPager());
83          return *this;
84      }
85  };
86
```

```
87  int main(void) {
88      //test the function in the class
89      string name1{ "lilei" }, number1{ "123456" }, name2{ "hanmeimei" }, number2{ "654321" };
90      BusinessTraveler b1{ name1,number1 };//test the constructor that takes two string arguments
91      cout << b1.BusinessTravelerToString() << '\n';
92      BusinessTraveler b2{ name2 }; //test the constructor that takes a string argument
93      cout << b2.BusinessTravelerToString() << '\n';
94      Pager p2{ number2 };//test the constructor that takes a string argument
95      b2.setPagerOfBusinessTraveler(p2);
96      cout << b2.BusinessTravelerToString() << '\n';
97      b2 = b1; //test the overloading assignment operator function
98      cout << b2.BusinessTravelerToString() << '\n';
99      return 0;
100 }
```

Output:

Answer2:

Code:

```cpp
#include<iostream>
#include<ctime>
 using std::cout;

class father {
private:
public:
    int a{ 0 };
    father() = default;
    virtual void vfprint(int i) {
        a = i;
    }
    void ifprint(int i) {
        a = i;
    }
};

class son :public father {
private:
public:
    int b{ 0 };
    son() = default;
    void vfprint(int i)override {
        b = i;
    }
};
```

```cpp
int main(void) {
    clock_t start1, start2, end1, end2;
    double duration1, duration2;
    son* s1 = new son();
    father* f1=dynamic_cast<father*>(s1);

    start1 = clock();
    for(int i=0;i< 1000000000;i++)
        f1->vfprint(i);
    s1->a = s1->b;
    end1 = clock();
    duration1 = (double)(static_cast<double>(end1) - static_cast<double>(start1)) / CLK_TCK;
    cout << "invoking virtual function \"vfprint()\" for 1000000000 times takes " << duration1 << " seconds!\n";

    start2 = clock();
    for (int i = 0; i < 1000000000; i++)
        f1->ifprint(i);
    s1->a = s1->b;
    end2 = clock();
    duration2 = (double)(static_cast<double>(end2) - static_cast<double>(start2)) / CLK_TCK;
    cout << "invoking non-virtual function \"ifprint()\" for 1000000000 times takes " << duration2 << " seconds!\n";

    return 0;
}
```

Output: