Software Technical Assessment

Question 1

Consider the following C++ program

```
int main()
{
    int x, y, z;
    x = 9;
    z = x + 1 + 1 * 0;
    y = z++;

    return 0;
}
```

1. What are the final values of x, y and z?

Question 2

Consider the following two C++ programs:

Program 1:

```
void incrementFunction(int input)
{
    input = input+1;
}

int main()
{
    int counter = 0;
    incrementFunction(counter);
    cout << "counter value is:"
    cout << counter;
    return 0;
}
```

Program 2:

```
void incrementFunction(int &input)
{
    input = input+1;
}

int main()
{
    int counter = 0;
    incrementFunction(counter);
    cout << "counter value is:"
    cout << counter;
    return 0;
}
```

1. Explain in few words what is the difference between the two programs?
2. What will be the final value of the counter variable?

Question 3

We are in 2035, a car rental company provides several types of vehicles (Petrol, Electric and Hybrid) for rental and they offer discounts for Electric and Hybrid cars.

- The basic rental price (before any deduction) is calculated as follows:

$$Basic\ Rental\ Price = Price\ per\ day * duration\ of\ the\ rental$$

- The Price per day is fixed to 50$.
- For Electric car, 50% discount on the price is offered.
- For Hybrid car, 25% discount on the price is offered

1. Write a code which allows to calculate the rental price of several vehicles. This shal be done by performing the following:
    - Define a class “Car” with a constructor that initialize PricePerDay(x) and RentalDuration(y)
    - Defines three subclasses “PetrolCar” and “ElectricCar” and “HybridCar”.
    - Defines the method getRentalPrice() in the three subclasses, which calculates the rental price.

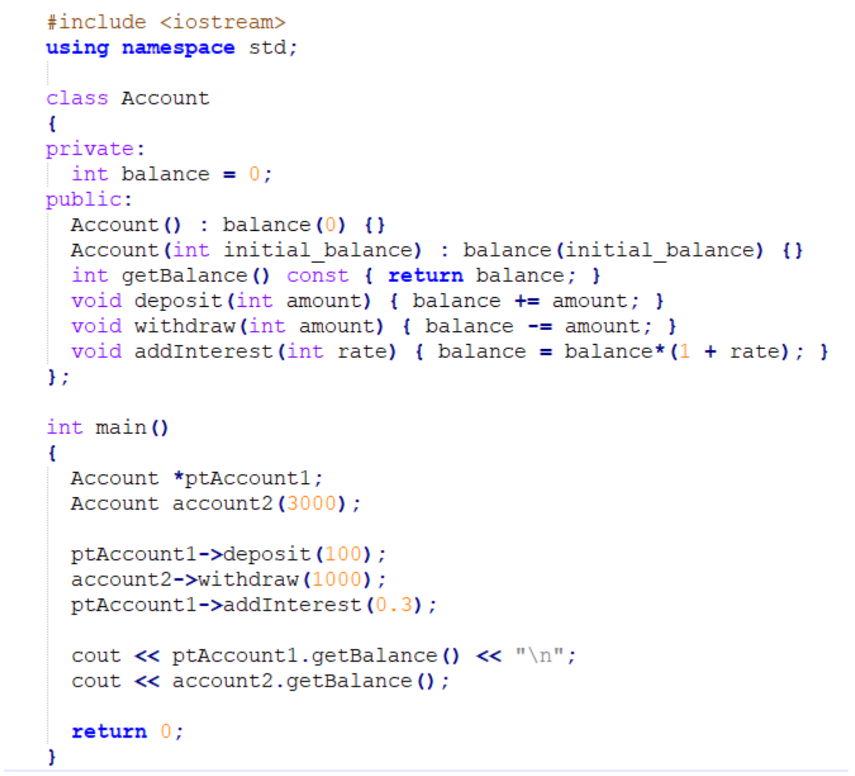
    You code shall work with the following main() function:

```
int main ()
{
  PetrolCar petrolCar(50,7);
  ElectricCar electricCar(50,10);
  HybridCar hybridCar(50,5);
  cout << petrolCar.getRentalPrice() << endl;
  cout << electricCar.getRentalPrice() << endl;
  cout << hybridCar.getRentalPrice() << endl;
  return 0;
}
```

2. What will be the outputted values (petrolCar.getRentalPrice(), electricCar.getRentalPrice() and hybridCar.getRentalPrice())?

Question 4

1. Perform a code review on the following C++ program. Note the errors and share your observations:

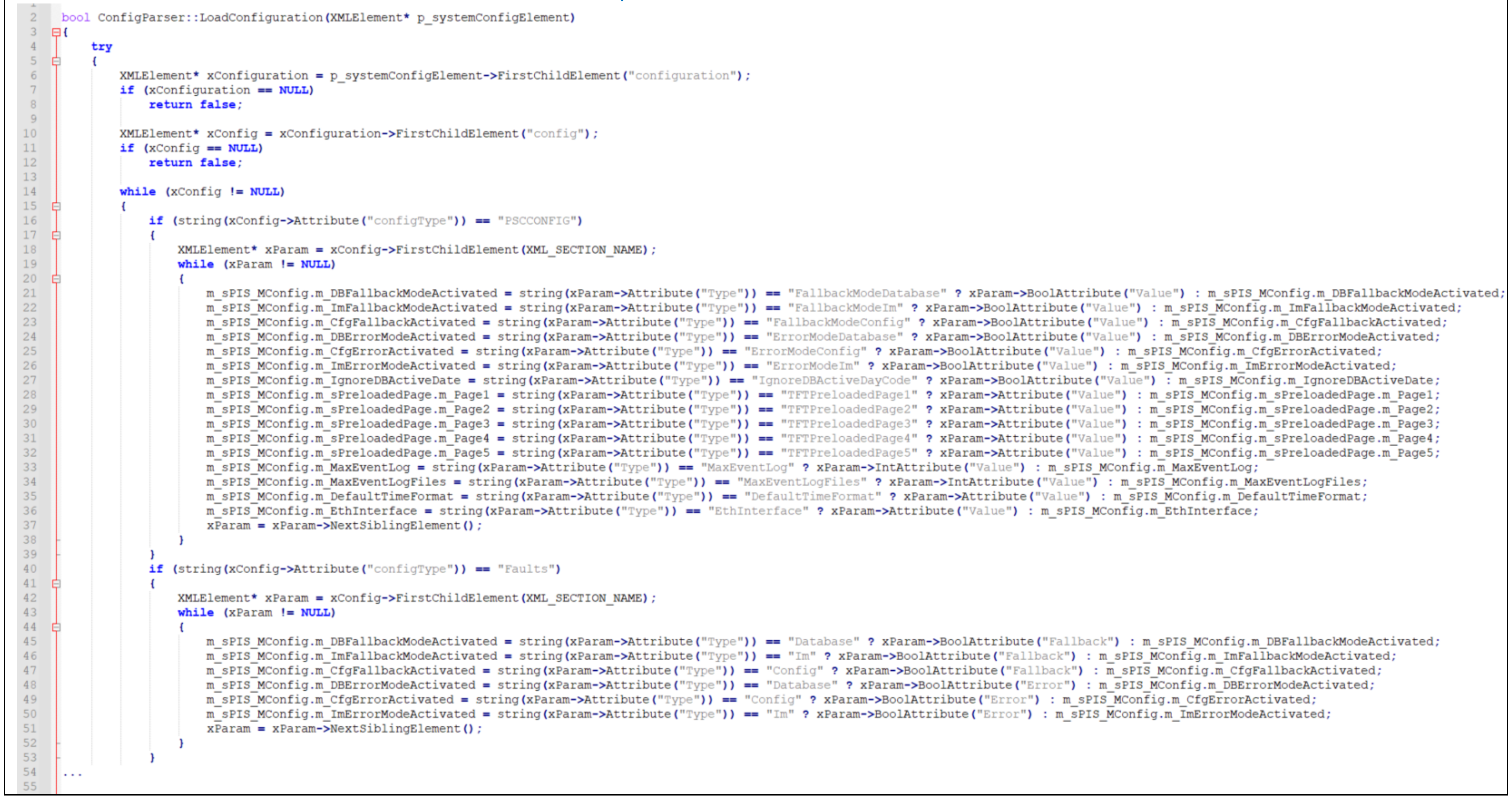


```cpp
#include <iostream>
using namespace std;

class Account
{
private:
  int balance = 0;
public:
  Account() : balance(0) {}
  Account(int initial_balance) : balance(initial_balance) {}
  int getBalance() const { return balance; }
  void deposit(int amount) { balance += amount; }
  void withdraw(int amount) { balance -= amount; }
  void addInterest(int rate) { balance = balance*(1 + rate); }
};

int main()
{
  Account *ptAccount1;
  Account account2(3000);

  ptAccount1->deposit(100);
  account2->withdraw(1000);
  ptAccount1->addInterest(0.3);

  cout << ptAccount1.getBalance() << "\n";
  cout << account2.getBalance();

  return 0;
}
```

2. Refactor the code and fix the errors you identified in 1).

Question 5

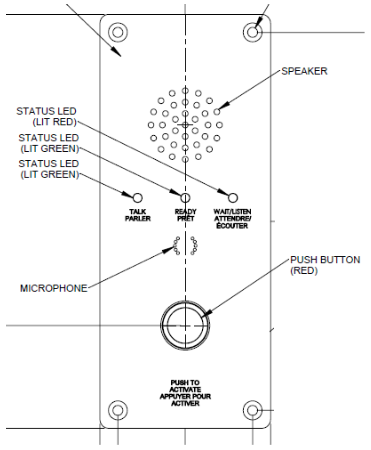
1. The following code shows an excerpt of a method used for reading a configuration from an XML file. Explain in your words why this is not considered clean code and how can this be improved.



```cpp
bool ConfigParser::LoadConfiguration(XMLElement* p_systemConfigElement)
{
    try
    {
        XMLElement* xConfiguration = p_systemConfigElement->FirstChildElement("configuration");
        if (xConfiguration == NULL)
            return false;

        XMLElement* xConfig = xConfiguration->FirstChildElement("config");
        if (xConfig == NULL)
            return false;

        while (xConfig != NULL)
        {
            if (string(xConfig->Attribute("configType")) == "PSCCONFIG")
            {
                XMLElement* xParam = xConfig->FirstChildElement(XML_SECTION_NAME);
                while (xParam != NULL)
                {
                    m_sPIS_MConfig.m_DBFallbackModeActivated = string(xParam->Attribute("Type")) == "FallbackModeDatabase" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_DBFallbackModeActivated;
                    m_sPIS_MConfig.m_ImFallbackModeActivated = string(xParam->Attribute("Type")) == "FallbackModeIm" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_ImFallbackModeActivated;
                    m_sPIS_MConfig.m_CfgFallbackActivated = string(xParam->Attribute("Type")) == "FallbackModeConfig" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_CfgFallbackActivated;
                    m_sPIS_MConfig.m_DBErrorModeActivated = string(xParam->Attribute("Type")) == "ErrorModeDatabase" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_DBErrorModeActivated;
                    m_sPIS_MConfig.m_CfgErrorActivated = string(xParam->Attribute("Type")) == "ErrorModeConfig" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_CfgErrorActivated;
                    m_sPIS_MConfig.m_ImErrorModeActivated = string(xParam->Attribute("Type")) == "ErrorModeIm" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_ImErrorModeActivated;
                    m_sPIS_MConfig.m_IgnoreDBActiveDate = string(xParam->Attribute("Type")) == "IgnoreDBActiveDayCode" ? xParam->BoolAttribute("Value") : m_sPIS_MConfig.m_IgnoreDBActiveDate;
                    m_sPIS_MConfig.m_sPreloadedPage.m_Page1 = string(xParam->Attribute("Type")) == "TFTPreloadedPage1" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_sPreloadedPage.m_Page1;
                    m_sPIS_MConfig.m_sPreloadedPage.m_Page2 = string(xParam->Attribute("Type")) == "TFTPreloadedPage2" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_sPreloadedPage.m_Page2;
                    m_sPIS_MConfig.m_sPreloadedPage.m_Page3 = string(xParam->Attribute("Type")) == "TFTPreloadedPage3" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_sPreloadedPage.m_Page3;
                    m_sPIS_MConfig.m_sPreloadedPage.m_Page4 = string(xParam->Attribute("Type")) == "TFTPreloadedPage4" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_sPreloadedPage.m_Page4;
                    m_sPIS_MConfig.m_sPreloadedPage.m_Page5 = string(xParam->Attribute("Type")) == "TFTPreloadedPage5" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_sPreloadedPage.m_Page5;
                    m_sPIS_MConfig.m_MaxEventLog = string(xParam->Attribute("Type")) == "MaxEventLog" ? xParam->IntAttribute("Value") : m_sPIS_MConfig.m_MaxEventLog;
                    m_sPIS_MConfig.m_MaxEventLogFiles = string(xParam->Attribute("Type")) == "MaxEventLogFiles" ? xParam->IntAttribute("Value") : m_sPIS_MConfig.m_MaxEventLogFiles;
                    m_sPIS_MConfig.m_DefaultTimeFormat = string(xParam->Attribute("Type")) == "DefaultTimeFormat" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_DefaultTimeFormat;
                    m_sPIS_MConfig.m_EthInterface = string(xParam->Attribute("Type")) == "EthInterface" ? xParam->Attribute("Value") : m_sPIS_MConfig.m_EthInterface;
                    xParam = xParam->NextSiblingElement();
                }
            }
            if (string(xConfig->Attribute("configType")) == "Faults")
            {
                XMLElement* xParam = xConfig->FirstChildElement(XML_SECTION_NAME);
                while (xParam != NULL)
                {
                    m_sPIS_MConfig.m_DBFallbackModeActivated = string(xParam->Attribute("Type")) == "Database" ? xParam->BoolAttribute("Fallback") : m_sPIS_MConfig.m_DBFallbackModeActivated;
                    m_sPIS_MConfig.m_ImFallbackModeActivated = string(xParam->Attribute("Type")) == "Im" ? xParam->BoolAttribute("Fallback") : m_sPIS_MConfig.m_ImFallbackModeActivated;
                    m_sPIS_MConfig.m_CfgFallbackActivated = string(xParam->Attribute("Type")) == "Config" ? xParam->BoolAttribute("Fallback") : m_sPIS_MConfig.m_CfgFallbackActivated;
                    m_sPIS_MConfig.m_DBErrorModeActivated = string(xParam->Attribute("Type")) == "Database" ? xParam->BoolAttribute("Error") : m_sPIS_MConfig.m_DBErrorModeActivated;
                    m_sPIS_MConfig.m_CfgErrorActivated = string(xParam->Attribute("Type")) == "Config" ? xParam->BoolAttribute("Error") : m_sPIS_MConfig.m_CfgErrorActivated;
                    m_sPIS_MConfig.m_ImErrorModeActivated = string(xParam->Attribute("Type")) == "Im" ? xParam->BoolAttribute("Error") : m_sPIS_MConfig.m_ImErrorModeActivated;
                    xParam = xParam->NextSiblingElement();
                }
            }
...
```

Question 6

Consider the following technical description of an audio device, called the “Passenger Call Unit (PCU)” device:
The “PCU” device provides communication between passengers and the driver in case of emergency in the train. To initiate the communication, the Push button must be pressed by the passenger. When this happens, an audio tone starts being played on the speaker and the “Wait/Listen” Indicator turns ON. This notifies the passenger that the call has been initiated, he can therefore wait until the call is answered by the driver of the train.
When this call is answered by the driver, the audio tone stops being played on the speaker. In addition, the “Talk” indicator turns ON and the “Wait/Listen” Indicator turns OFF. This indicates to the passenger that he can now speak with the driver. Audio from passenger is captured by the Microphone.
When the driver is speaking, the “Talk” indicator turns OFF and the “Wait/Listen” Indicator turns ON in order to indicate to the passenger that he should listen, in this case, the audio coming from the driver can be heard over the speaker.
The call can only be terminated by the driver, when this happens, both “Talk” and “Wait/Listen” Indicators turn OFF.



1. Create the state machine diagram describing the states of the “PCU” device. Please describe each of the state and the transitions.
2. Write a code (C or C++) allowing to implement this state machine.