

Section 6.1 Answers to Odd Numbered Questions

1) Dinosaur barney;

```
barney.name = "T-Rex";
barney.weight = 10000.0;
barney.length = 35.0;
barney.carnivorous = true;
```

3) Dinosaur barney;

```
char eatsMeat;

cout << "Dinosaur's name:   ";
getline(cin, barney.name);
cout << "Dinosaur's weight: ";
cin >> barney.weight;
cout << "Dinosaur's length: ";
cin >> barney.length;
cout << "Eat's meat? (y/n): ";
cin >> eatsMeat;
if (eatsMeat == 'y' || eatsMeat == 'Y')
    barney.carnivorous = true;
else
    barney.carnivorous = false;
```

```
5) cout << "Name:           " << barney.name << endl
    << "Weight:           " << barney.weight << " pounds" << endl
    << "Length:           " << barney.length << " feet" << endl
    << "Canivorous: ";
if (barney.carnivourous)
    cout << "Yes" << endl;
else
    cout << "No" << endl;
```

7) T-Rex

```
9) bool DinoMatch(Dinosaur lizard1, Dinosaur lizard2)
{
    if (lizard1.name == lizard2.name)
        return true;
    return false;
}
```

```
11) struct Student
    {
        string name;
        string id;
        double exam1;
        double exam2;
        double exam3;
    };

13) struct Truck
    {
        string id;
        int    purchased;
        string license;
        string vin;
    };
```

Section 6.2 and 7.1 Answers to Odd Numbered Questions

```
1) Puzzle p;
   p.MoveSquireDown('B');
   p.MoveSquareRight('A');
   p.MoveSquareUp('C');
   p.MoveSquareLeft('B');
   p.MoveSquareDown('A');

3) LaCucaracha bug;
   bug.StompBug();
   if (bug.IsAlive())
   {
       bug.ChangeDirection();
       bug.RunLikeHeck();
   }
```

```
5) LaCucaracha bug;
while(bug.IsAlive())
{
    if (bug.IsLight())
    {
        bug.StompBug();
        if (bug.IsAlive())
            bug.RunLikeHeck();
    }
    else
    {
        if (bugIsFood())
            bug.Eat();
        else
        {
            bug.ChangeDirection();
            bug.RunLikeHeck();
        }
    }
}

7) class Employee
{
public:
    Employee();
    Employee(string initName, string initId);
    void SetRate(double initRate);
    void SetHours(double initHours);
    double CalcPay();
private:
    string name, id;
    double rate, hours;
};
```

```
9) Employee::Employee()
{
    name = "nobody";
    id = -1;
    rate = 0;
    hours = 0;
}

Employee::Employee(string initName, string initId)
{
    name = initName;
    id = initId;
    rate = 0;
    hours = 0;
}

void Employee::SetRate(double initRate)
{
    rate = initRate;
}

void Employee::SetHours(double initHours)
{
    hours = initHours;
}

double Employee::CalcPay()
{
    return rate * hours;
}

void main()
{
    Employee e;
}
```

Programming Project: Candy Machine

```
// *****
// Specification File -- machine.h
// *****

#include <string>

class CandyMachine
{
public:
    CandyMachine();
    ~CandyMachine();
    void Menu();
    int GetChoice();
    int Process(int);
    void BuyPC();
    void BuyJB();
    void BuyCCC();
    void BuyVCG();
    int GetMoney();
    int CheckStock(int);
    int ProcessMgr();
    int VerifyPW();
    void CheckMoney();
    void ChangePW();
    void Restock();
private:
    int pc, jb, ccc, vcg;
    string pw;
    float price, money;
};

// *****
// Implementation File -- machine.cc
// *****

#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include "machine.h"

CandyMachine::CandyMachine()
{
    ifstream inFile;
```

```

inFile.open("pa8.dat");

if (!inFile)
{
    pc = 20;
    jb = 20;
    ccc = 20;
    vcg = 20;
    price = 0.50;
    money = 0.0;
    pw = "password";
}
else
{
    inFile >> pc >> jb >> ccc >> vcg >> money >> pw;
    price = 0.50;
    inFile.close();
}
}

CandyMachine::~CandyMachine()
{
    ofstream outFile;
    outFile.open("pa8.dat");
    outFile << pc << endl << jb << endl << ccc << endl
        << vcg << endl << money << endl << pw << endl;
    outFile.close();
}

void CandyMachine::Menu()
{
    cout<<"\n\nWelcome to Nona's Candy Machine\n\n";
    cout<<"Select one of the following:\n";
    cout<<" 1 - Potato Chips\n";
    cout<<" 2 - Jaw Breakers\n";
    cout<<" 3 - Chocolate Chip Cookies\n";
    cout<<" 4 - Vanilla Chewing Gum\n\n";
}

int CandyMachine::GetChoice()
{
    int choice;
    cout<<"Choice: ";
    cin>>choice;
    return choice;
}

int CandyMachine::Process(int choice)
{

```

```

int     status = 0;
string  xtra;
switch(choice)
{
    case 1:  BuyPC();
             break;
    case 2:  BuyJB();
             break;
    case 3:  BuyCCC();
             break;
    case 4:  BuyVCG();
             break;
    case 5:  status = ProcessMgr();
             break;
    default: cout<<"Invalid Choice -- Try Again\n"
             <<"Press C and then <Enter> to Continue . . .\n";
             cin>>xtra;
}
return status;
}

int CandyMachine::VerifyPW()
{
    string mgrPW, xtra;
    cout<<"This option requires a password.\n";
    cout<<"Please enter password:  ";
    cin>>mgrPW;
    if (mgrPW == pw)
        return 1;
    cout<<"Invalid Password -- Access Denied!\n";
    cout<<"Press C and then <Enter> to Continue . . .\n";
    cin>>xtra;
    return -1;
}

int CandyMachine::ProcessMgr()
{
    int     choice;
    string  xtra;
    if (VerifyPW() == -1)        // Invalid Password
        return 0;
    do
    {
        cout<<"\n\nWelcome to the Manager's Menu\n\n";
        cout<<"Select one of the following:\n";
        cout<<"  1  -  Check Total Money Collected\n";
        cout<<"  2  -  Change Password\n";
        cout<<"  3  -  Restock Machine\n";
        cout<<"  4  -  Shut Down the Machine\n";
        cout<<"  5  -  Return to Main Menu\n\n";
    }
}

```

```

    cout<<"Choice:  ";
    cin>>choice;
    switch(choice)
    {
        case 1:  CheckMoney();
                break;
        case 2:  ChangePW();
                break;
        case 3:  Restock();
                break;
        case 4:  cout<<"Shut Down Selected !!!!\n"
                <<"Good Bye . . . . .\n\n\n";
                return 99;
        case 5:  return 0;
        default: cout<<"Invalid Choice -- Try Again\n"
                <<"Press C and then <Enter> to Continue . . .\n";
                cin>>xtra;
    }

} while (choice != 5);
}

void CandyMachine::BuyPC()
{
    if (GetMoney() == 1 && CheckStock(1) == 1)
    {
        pc--;
        money = money + 0.50;
    }
}

void CandyMachine::BuyJB()
{
    if (GetMoney() == 1 && CheckStock(2) == 1)
    {
        jb--;
        money = money + 0.50;
    }
}

void CandyMachine::BuyCCC()
{
    if (GetMoney() == 1 && CheckStock(3) == 1)
    {
        ccc--;
        money = money + 0.50;
    }
}

```

```

}

void CandyMachine::BuyVCG()
{
    if (GetMoney() == 1 && CheckStock(4) == 1)
    {
        vcg--;
        money = money + 0.50;
    }
}

int CandyMachine::GetMoney()
{
    float moneyIn;
    string xtra;
    cout<<"Please enter your money: ";
    cin>>moneyIn;
    if (moneyIn < price)
    {
        cout<<"Insufficient money -- sale cancelled!\n";
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return -1;
    }
    else
    {
        if (moneyIn > price)
            cout<<"Your change is: $"<<moneyIn - price<<endl;
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return 1;
    }
}

int CandyMachine::CheckStock(int choice)
{
    string xtra;
    switch(choice)
    {
        case 1: if (pc < 1)
                {
                    cout<<"Potato Chips Out of Stock -- sale cancelled!\n";
                    cout<<"Press C and then <Enter> to Continue . . .\n";
                    cin>>xtra;
                    return -1;
                }
                break;
        case 2: if (jcb < 1)
                {

```

```

        cout<<"Jaw Breakers Out of Stock -- sale cancelled!\n";

        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return -1;
    }
    break;
case 3:  if (ccc < 1)
    {
        cout<<"Cookies Out of Stock -- sale cancelled!\n";
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return -1;
    }
    break;
case 4:  if (vcg < 1)
    {
        cout<<"Chewing Gum Out of Stock -- sale cancelled!\n";
        cout<<"Press C and then <Enter> to Continue . . .\n";
        cin>>xtra;
        return -1;
    }
    break;
}
return 1; // Selection is available
}

void CandyMachine::CheckMoney()
{
    string xtra;
    cout<<"There is $"<<money<<" in the machine.\n";
    cout<<"Press C and then <Enter> to Continue . . .\n";
    cin>>xtra;
}

void CandyMachine::ChangePW()
{
    string xtra;
    cout<<"Enter the new password:  ";
    cin>>pw;
    cout<<"Password has been changed!\n";
    cout<<"Press C and then <Enter> to Continue . . .\n";
    cin>>xtra;
}

void CandyMachine::Restock()
{
    string xtra;
    pc = 20;
    jb = 20;
}

```

```
    ccc = 20;
    vcg = 20;
    money = 0.0;
    cout<<"Machine has been restocked!\n";
    cout<<"Press C and then <Enter> to Continue . . .\n";
    cin>>extra;
}
```

```
// *****
//      Client Code -- candy.cc
// *****

#include <string>
#include "machine.h"
using namespace std;

int main()
{
    CandyMachine myCM;
    int choice;
    int status = 0;

    do
    {
        myCM.Menu();
        choice = myCM.GetChoice();
        status = myCM.Process(choice);
        if (status == 99)    // Manager chose to Shut Down
            break;
    } while (status == 0);

    return 0;
}
```