WITH Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO, TicketProgramPackage;

USE Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO, TicketProgramPackage;

PROCEDURE TicketProgram IS

MenuChoice : Natural := 0;

NewburgCost : CONSTANT Float := 3.50;

NyackCost : CONSTANT Float := 5.25;

NewburgSupply : Integer := 10;

NyackSupply : Integer := 15;

NumberTickets : Integer;

TotalCost : Float := 0.00;

AmountPaid :Float := 0.00;

TotalChange : Float := 0.00;

TotalProfit : Float := 0.00;

PROCEDURE Newburg IS

BEGIN

Put("Purchase how many tickets to Newburg?");

New\_Line;

Put(">");

Get(NumberTickets);

IF NewburgSupply - NumberTickets >= 0 AND NumberTickets > 0 THEN

New\_Line;

Put("Sufficient Supply");

TotalCost := NewburgCost \* Float(NumberTickets);

New\_Line;

Put("The total cost is > $");

Put(TotalCost,Aft=>2,Exp=>0);

New\_Line;

Put ("Enter amount paid");

New\_Line;

Put("> $");

Get(AmountPaid);

TotalChange := AmountPaid - TotalCost;

IF TotalChange >= 0.0 THEN

Find\_Change(TotalChange);

NewburgSupply := NewburgSupply - NumberTickets;

TotalProfit := TotalProfit + TotalCost;

ELSE

Put("Insufficient Payment: Transaction Terminated");

END IF;

ELSIF NumberTickets <= 0 THEN

New\_Line;

Put("Invalid Input");

ELSE

New\_Line;

Put("Insufficient Supply");

Check\_Tickets(NewburgSupply,NyackSupply);

END IF;

END Newburg;

PROCEDURE Nyack IS

BEGIN

Put("Purchase how many tickets to Nyack?");

New\_Line;

Put(">");

Get(NumberTickets);

IF NyackSupply - NumberTickets >= 0 AND NumberTickets > 0 THEN

New\_Line;

Put("Sufficient Supply");

TotalCost := NyackCost \* Float(NumberTickets);

New\_Line;

Put("The total cost is > $");

Put(TotalCost,Aft=>2,Exp=>0);

New\_Line;

Put ("Enter amount paid");

New\_Line;

Put("> $");

Get(AmountPaid);

TotalChange := AmountPaid - TotalCost;

IF TotalChange >= 0.0 THEN

Find\_Change(TotalChange);

NyackSupply := NyackSupply - NumberTickets;

TotalProfit := TotalProfit + TotalCost;

ELSE

Put("Insufficient Payment: Transaction Terminated");

END IF;

ELSIF NumberTickets <= 0 THEN

New\_Line;

Put("Invalid Input");

ELSE

New\_Line;

Put("Insufficient Supply");

Check\_Tickets(NewburgSupply,NyackSupply);

END IF;

END Nyack;

PROCEDURE TrainDeparture IS

TYPE Trains IS (nyack, newburg, quit, nothing);

PACKAGE Trains\_IO IS NEW Ada.Text\_IO.Enumeration\_IO(Enum=>Trains);

SubmenuChoice : Trains := nothing;

BEGIN

WHILE SubmenuChoice /= quit LOOP

Put("Train Departure Submenu:");

New\_Line;

Put("Type 'newburg' to process Train Departure to Newburg");

New\_Line;

Put("Type 'nyack' to process Train Departure to Nyack");

New\_Line;

Put("Type 'quit' to return to Main Menu");

New\_Line;

Put(">");

Trains\_IO.Get(SubmenuChoice);

IF SubmenuChoice = newburg

THEN

NewburgSupply := 10;

ELSIF SubmenuChoice = nyack

THEN

NyackSupply := 15;

ELSIF SubmenuChoice = Quit

THEN

NULL;

ELSE

New\_Line;

Put("Invalid Input: Reenter");

New\_Line;

New\_Line;

END IF;

END LOOP;

END TrainDeparture;

BEGIN

WHILE MenuChoice /= 6 LOOP

New\_Line;

Put("Main Menu:");

New\_Line;

Put("1)Sell Tickets to Newburg");

New\_Line;

Put("2)Sell Tickets to Nyack");

New\_Line;

Put("3)Process Train Departure");

New\_Line;

Put("4)Check Tickets Remaining");

New\_Line;

Put("5)View Total Sales");

New\_Line;

Put("6)End Program");

New\_Line;

Put(">");

Get(MenuChoice);

IF MenuChoice = 1

THEN

New\_Line;

Newburg;

New\_Line;

ELSIF MenuChoice = 2

THEN

New\_Line;

Nyack;

New\_Line;

ELSIF MenuChoice = 3

THEN

New\_Line;

TrainDeparture;

New\_Line;

ELSIF MenuChoice = 4

THEN

New\_Line;

Check\_Tickets(NewburgSupply,NyackSupply);

New\_Line;

ELSIF MenuChoice = 5

THEN

New\_Line;

View\_Total\_Sales(TotalProfit);

New\_Line;

END IF;

END LOOP;

END TicketProgram;

PACKAGE TicketProgramPackage IS

PROCEDURE View\_Total\_Sales (TotalProfit : IN Float);

--Pre: Inputs Total Profit

--Post: Outputs the total profit to the user

PROCEDURE Check\_Tickets (NewburgSupply : IN Integer; NyackSupply : IN Integer);

--Pre: Input Integer values of the number of Newburg and Nyack tickets available

--Post: Lists out the tickets available

PROCEDURE Find\_Change (TotalChange : IN Float);

--Pre: Input amount of change

--Post: Outputs the number of each increment of change that must be returned

END TicketProgramPackage;

WITH Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO;

USE Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO;

PACKAGE BODY TicketProgramPackage IS

PROCEDURE View\_Total\_Sales (TotalProfit : IN Float) IS

BEGIN

New\_Line;

Put("The total of sales is $");

Put(TotalProfit,Aft=>2,Exp=>0);

END View\_Total\_Sales;

PROCEDURE Check\_Tickets (NewburgSupply : IN Integer; NyackSupply : IN Integer) IS

Counter : Integer;

BEGIN

New\_Line;

Counter := 1;

Put("Ticket Inventory:");

New\_Line;

WHILE NewburgSupply - Counter >= 0 LOOP

Put(Counter);

Put(" Newburg Ticket");

New\_Line;

Counter := Counter +1;

END LOOP;

New\_Line;

Counter := 1;

FOR Count IN 1..NyackSupply LOOP

Put(Count);

Put(" Nyack Ticket");

New\_Line;

END LOOP;

END Check\_Tickets;

PROCEDURE Find\_Change (TotalChange: IN Float) IS

TotalChangeHundred : Float;

TotalChangeInteger : Integer;

NumberDollars : Integer;

NumberQuarters : Integer;

NumberDimes : Integer;

NumberNickels : Integer;

NumberPennies : Integer;

FUNCTION Find\_Number\_Coins(Value,Amount:Integer)RETURN Integer IS

Number : Natural := 0;

Amount\_Internal : Integer;

BEGIN

Amount\_Internal := Amount;

WHILE Value <= Amount\_Internal LOOP

Number := Number + 1;

Amount\_Internal := Amount\_Internal - Value;

END LOOP;

RETURN Number;

END Find\_Number\_Coins;

BEGIN

--Initialize variable;

NumberDollars := 0;

NumberQuarters := 0;

NumberDimes := 0;

NumberNickels := 0;

NumberPennies := 0;

--convert TotalChange to TotalChangeInteger

TotalChangeHundred := TotalChange \* 100.0;

TotalChangeInteger := Integer(TotalChangeHundred);

--calculate amount of dollars needed

NumberDollars := Find\_Number\_Coins(100,TotalChangeInteger);

TotalChangeInteger := TotalChangeInteger - (NumberDollars \* 100);

NumberQuarters := Find\_Number\_Coins(25,TotalChangeInteger);

TotalChangeInteger := TotalChangeInteger - (NumberQuarters \* 25);

NumberDimes := Find\_Number\_Coins(10,TotalChangeInteger);

TotalChangeInteger := TotalChangeInteger - (NumberDimes \* 10);

NumberNickels := Find\_Number\_Coins(5,TotalChangeInteger);

TotalChangeInteger := TotalChangeInteger - (NumberNickels \* 5);

NumberPennies := Find\_Number\_Coins(1,TotalChangeInteger);

TotalChangeInteger := TotalChangeInteger - (NumberPennies \* 1);

--Output Results

New\_Line;

Put("Give the customer ");

Put(NumberDollars);

Put(" Dollars");

New\_Line;

Put("Give the customer ");

Put (NumberQuarters);

Put(" Quarters");

New\_Line;

Put("Give the customer ");

Put (NumberDimes);

Put(" Dimes");

New\_Line;

Put("Give the customer ");

Put (NumberNickels);

Put(" Nickels");

New\_Line;

Put("Give the customer ");

Put (NumberPennies);

Put (" Pennies");

END Find\_Change;

End TicketProgramPackage;