```
    1: //  Author: Wayne Buchner
    2: //  Student ID: 6643140
    3: //  Date: 20 04 2011
    4: //  Program: Calculate Change Assignment 6 Program
    5: // Description: This program initialises a set of coins, accepts a value for a purch
ase
    6: //                  and tests whether enough change is present and if is able to ret
urn the change given
    7: //              in the correct demonination.
    8:
    9: {$MODE objfpc}
   10: program ChangeCalculator;
   11: uses SysUtils, math;
   12:
   13: //  Type Declarations.
   14: type
   15:     CoinData = record
   16:         value           :   Integer;
   17:         count           :   Integer;
   18:         description     :   String;
   19:     end;
   20:
   21: //  Declare the CoinSet array here...
   22: CoinSet = array [0..5] of CoinData;
   23:
   24: //  Delcare the CombineCoinsOp enumeration here...
   25: CombineCoinsOp = (AddCoins, RemoveCoins);
   26:
   27:
   28: //  Zero all of the counts in a set of coins
   29: procedure ZeroCounts(var aSet: CoinSet);
   30: var
   31:     i : Integer;
   32: begin
   33:   for i := Low(aSet) to High(aSet) do
   34:         begin
   35:             aSet[i].count := 0;
   36:         end;
   37: end;
   38:
   39: //  Setup the coin data, initialise value, description, and count (set to 0)
   40: procedure InitialiseCoins(out aSet: CoinSet);
   41: begin
   42:   aSet[0].value := 5;
   43:   aSet[0].description := '5c';
   44:   aSet[1].value := 10;
   45:   aSet[1].description := '10c';
   46:   aSet[2].value := 20;
   47:   aSet[2].description := '20c';
   48:   aSet[3].value := 50;
   49:   aSet[3].description := '50c';
   50:   aSet[4].value := 100;
   51:   aSet[4].description := '$1';
   52:   aSet[5].value := 200;
   53:   aSet[5].description := '$2';
   54:
   55:   ZeroCounts(aSet);
   56: end;
   57:
   58: //  Draws a horizontal line on the Console
   59: procedure DrawLine();
   60: var
   61:   i: Integer;
   62: begin
   63:   for i := 0 to 70 do
   64:   begin
   65:     Write('-');
   66:   end;
   67:   WriteLn();
   68: end;
   69:
   70: //  Prints a message followed by the coin data from a set of coins.
   71: procedure PrintCoins(message: String; const aSet: CoinSet);
   72: var
   73:     i : Integer;
   74: begin
   75:     Write(message,' ');
   76:     for i := Low(aSet) to High(aSet) do
   77:         begin
   78:         Write(aSet[i].count, ' x ', aSet[i].description);
   79:         if i < High(aSet) then
   80:             begin
   81:                 Write(', ');
   82:             end;
   83:         end;
   84:         WriteLn();
   85: end;
   86:
   87: //  Displays the coins in the machine between two horizontal lines.
   88: procedure PrintMachineStatus(const machineCoins: CoinSet);
   89: begin
   90:   DrawLine();
   91:   PrintCoins('In Machine - ', machineCoins);
   92:   DrawLine();
   93:   WriteLn();
   94: end;
   95:
   96: //  Reads in an integer, using the message as the prompt for the value
   97: function ReadInteger(message: String): Integer;
   98: var
   99:   temp: String;
  100: begin
  101:   Write(message);
  102:   ReadLn(temp);
  103:
  104:   while not TryStrToInt(temp, result) do
  105:   begin
  106:     WriteLn('Please enter a whole number.');
  107:
  108:     Write(message);
  109:     ReadLn(temp);
  110:   end;
  111: end;
  112:
  113: //  Reads coin data from the user into a set of coins.
  114: procedure ReadCoinCounts(message: String; var intoSet: CoinSet);
  115: begin
  116:   Write(message, ' (5, 10, 20, 50, 100, 200): ');
  117:   ReadLn(intoSet[0].count,
  118:     intoSet[1].count,
  119:     intoSet[2].count,
  120:     intoSet[3].count,
  121:     intoSet[4].count,
  122:     intoSet[5].count
  123:     );
  124:   WriteLn();
```

```
  125: end;
  126:
  127: //  Calculates the total value of a set of coins.
  128: function TotalValue(const aSet: CoinSet): Integer;
  129: var
  130:     i : Integer;
  131: begin
  132:     result :=0;
  133:     for i := Low(aSet) to High(aSet) do
  134:         begin
  135:             result += (aSet[i].value * aSet[i].count);
  136:         end;
  137: end;
  138:
  139: // The function returns true if it was about to distribute the correct
  140: // change, otherwise it returns false.
  141: function CalculateChange(amount: Integer; const machineCoins: CoinSet; const fromCoi
ns: CoinSet; var intoCoins: CoinSet): Boolean;
  142: var
  143:     i, val : Integer;
  144: begin
  145:     result := False;
  146: for i := High(machineCoins) downto Low(machineCoins) do
  147:     begin
  148:
  149:     if machineCoins[i].count > 0 then
  150:     begin
  151:     val := amount DIV machineCoins[i].value;
  152:     if machineCoins[i].count <= val then
  153:         begin
  154:         intoCoins[i].count := machineCoins[i].count;
  155:         amount := amount - (machineCoins[i].count * machineCoins[i].value);
  156:         end
  157:     else
  158:         begin
  159:         intoCoins[i].count := val;
  160:         amount := amount - (val * machineCoins[i].value);
  161:         end;
  162:     end;
  163: end;
  164:     if amount = 0 then
  165:     begin
  166:         result := True;
  167:     end
  168: end;
  169:
  170: //  Adds or removes the coins in coinsDiff from the fromCoins.
  171: procedure CombineCoins(const coinDiff: CoinSet; var alterCoins: CoinSet; operation:
CombineCoinsOp);
  172: var
  173:     i : Integer;
  174: begin
  175:     for i:= low(coinDiff) to High(CoinDiff) do
  176:     if operation = AddCoins then
  177:     begin
  178:         alterCoins[i].count := alterCoins[i].count + coinDiff[i].count;
  179:     end
  180:     else if operation = RemoveCoins then
  181:     begin
  182:         alterCoins[i].count := alterCoins[i].count - coinDiff[i].count;
  183:     end;
  184: end;
  185:
  186: //  Performs a purchase involving determining cost,
  187: procedure PerformPurchase(var machineCoins: CoinSet; var purchaseCoins: CoinSet);
  188: var
  189:   cost, diff: Integer;
  190:   payment, change: CoinSet;
  191: begin
  192:     //  Incoming coins
  193:     InitialiseCoins(payment);
  194:
  195:     InitialiseCoins(change);
  196:
  197:     cost := ReadInteger('PP Cost in cents: ');
  198:     ReadCoinCounts('PP Coins used', payment);
  199:
  200:     diff := TotalValue(payment) - cost;
  201:     WriteLn('Change Required ', diff,' cents' );
  202:     // put coins paid into machine
  203:     CombineCoins(payment, machineCoins, AddCoins);
  204:
  205:     //  CALLING CALCULATE CHANGE HERE  TRUE or FALSE
  206:     if CalculateChange(diff, machineCoins, purchaseCoins, change) then
  207:         begin
  208:         //  if calculatechange is true do this.....check this function with remove c
oins!
  209:         CombineCoins(change, machineCoins, RemoveCoins);
  210:         PrintCoins('Change required :', change);
  211:         end
  212:     else
  213:         begin
  214:         WriteLn('Not Enough Change Sorry');
  215:         CombineCoins(payment, machineCoins, RemoveCoins);
  216:         WriteLn(TotalValue(payment),' refunded');
  217:         end;
  218:     WriteLn();
  219: end;
  220:
  221: //  Main
  222: procedure Main();
  223: var
  224:   machineCoins, customerCoins: CoinSet;
  225:   i : Boolean;
  226: begin
  227:     i := True;
  228:     InitialiseCoins(machineCoins);
  229:
  230:     InitialiseCoins(customerCoins);
  231:     while i = True do
  232:     begin
  233:         ReadCoinCounts('Coins to add to machine: ',machineCoins);
  234:         CombineCoins(customerCoins, machineCoins, AddCoins);
  235:         PrintMachineStatus(machineCoins);
  236:         WriteLn('Total value of machine coins is: ', TotalValue(machineCoins));
  237:         PerformPurchase(machineCoins, customerCoins);
  238:         CombineCoins(customerCoins, machineCoins, RemoveCoins);
  239:         //  test remove coins
  240:         PrintMachineStatus(machineCoins);
  241:         WriteLn('Total value of machine coins is: ', TotalValue(machineCoins));
  242:     end;
  243: end;
  244:
  245: begin
  246:   Main();
  247: end.
```