This is perl documentation for MS4. This is for the user and computer friendly tables as well as code for the ultra regarding whether or not it is worthwhile to startup the power plant. Also this program connot be run unless our other perl program runs first, we removed the file splitting code for the National Weather Service text file. It will read the text files the other program makes.

print "What is the cost of your electricity?\n";        #query's user about cost of electricity  
$cost = <>;                            #input from the user  
print "Thank you.\n";  
Prompts user for electricity cost

$x = 0;                                #set counter x equal to 0  
$cdf[0] = 0;                            #set the array cdf[0] equal to 0

for ($z = 1; $z <= 13; $z++)                    #for loop for z, to keep track of the array   
 {  
  if ($z <= 2 || $z >= 11)                    #for the cdf values of 5, 10 and 95, 100       
   {  
    $cdf[$z] = $cdf[$x]+ 0.05;                    #makes the above cdf values 0.05  
    $x++                            #x increments, one less than z  
   }  
  else                                #for all others  
   {  
    $cdf[$z] = $cdf[$x]+ 0.10;                    #make all the other cdf values 0.10  
    $x++;                        #x increments, one less than z  
  }

}                   #closes the for loop, with the z counter  
Sets the cdf

do  
 {  
  print "What station model would you like to evaluate?\n";    #query's user about station model  
  $stationu = <>;                        #input from the user  
  chomp $stationu;  
  print "Thank you.\n";  
User input for station model

  $file6 = $stationu."\_peopletable."."txt";                #renames the file for each station  
  open (OUTFILE6, "> $file6") or die "Can't open $file1 for write: $!";    #opens output filehandle  
  $file7 = $stationu."\_ultra."."txt";                    #renames the file for each station  
  open (OUTFILE7, "> $file7") or die "Can't open $file1 for write: $!";    #opens output filehandle  
  $file8 = $stationu."\_computertable."."txt";                #renames the file for each station  
  open (OUTFILE8, "> $file8") or die "Can't open $file1 for write: $!"; #opens output filehandle  
Creates text files for tables and outputs, named by Station name.

  open (FH, $stationu.".txt") ||  
    die "ERROR Unable to open Dates: $!\n";                    #opens up textfile and assigns it a file handle  
  @array = <FH>;                        #puts the textfile into an array  
  close FH;                            #closes the file handle

  print OUTFILE6 "\t\t\t\t\t\t\tPRICES\n\n";   # header

  Creates a header for the top of the user friendly table

  for($i = 0; $i <= 74; $i++)                     #looping through the 75 rows  
   {                          
    @Temp = split(" ", @array[$i]);                #split the file array into a Temperature array   
Goes through each of the 75 lines in one Station model and splits them up into an array

if ($i %25 == 0)

{

  print OUTFILE6 " \n";    # header

  print OUTFILE6 "|  CDF 0 |  CDF 5 | CDF 10 | CDF 20 | CDF 30 | CDF 40 | CDF 50 | CDF 60 | CDF 70 | CDF 80 | CDF 90 | CDF 95 | CDF 100|\n";      
  print OUTFILE6 "'--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------'\n";

}

Prints a new CDF header to the user friendly output file every time the i counter is evenly divisible by 25

    for($p = 1; $p <= 13; $p++)                    #loop through the 13 values in the row  
     {  
      $price = sprintf("%.2f", 1.36 \*(abs($Temp[$p]-55)) +20);  #convert Temp to price  
      print OUTFILE6 "|  $price ";  
    print OUTFILE8 " $price ";                #output price to the table  
Calculates price to 2 decimal places for each temperature

if ($p == 13)              #every 13 values, aka end of row             
       {

    print OUTFILE8 "\n"; #computer friendly, new line  
        print OUTFILE6 "|"; #user friendly, bar  
        print OUTFILE6 "\n"; #user friendly, new line  
        print OUTFILE6 "'--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------'";  
        print OUTFILE6 "\n";                  # user friendly, new line  
       }  
Outfile6 prints into the text files with a border line in between for the setup of our user friendly tables. Outfile8 prints the temperatures into a file with 13 across, and all separated by a space, more computer friendly

      $Pindicator = ($price - $cost);                #calculates the profit

      $profit = sprintf("%.2f", $profit + $Pindicator \* $cdf[$p]); #calculates profit for the whole week      
      $worthit = sprintf("%.2f", -10000 + $profit); #determines if we overcome the startup cost  
      $plant = sprintf("%.2f", 10000 - $profit); #keeps track of how much until we overcome the startup cost  
      $xxxx = 3 \* $i; #calculates the number of hours elapsed   
     }

These are different profit calculations to two decimal places

    if($worthit > 0)   {                      #you have made a profit   
      print OUTFILE7 "To make $worthit profit your plant will have run for $xxxx hours. \n";  
     }  
    else   {                          #you have not yet made a profit  
      print OUTFILE7 "If the plant runs $xxxx hours, you will still need to make $plant to break even.\n";  
     }

}                                #closes the outside for loop, with the i counter  
Prints to a text file whether or not a profit is made. This does not speculate on whether it would be cost effective to operate the plant during the 11 day time period. This just gives the data that would help with that decision. It gives two outputs, if the accumulated profit has not yet reached the startup cost (of $10000), it tells the user how much more need to be made, otherwise it tells how much profit has been made.

print "Did you want to evaluate another Station(y/n)?\n"; #Input from the user to evaluate another station  
  $ans = <>;  
  chomp $ans;  
 }while ($ans eq 'y');       #ends the Do-While loop  
Prompts you to do another station calculation starting the beginning of the program

close OUTFILE6 or die "Can't close file: $!";            #closes the files used

close OUTFILE7 or die "Can't close file: $!";   
close OUTFILE8 or die "Can't close file: $!";

Close all of our open file handles and creates an error message if it could not complete it.