```
/*****************************************************************************

                            mangrove_madness.h

*****************************************************************************/

//The function that defines the energy required to run the mangrove filter (only the energy to run the pumps we's expect)
double energy (double pump_hp, double hrs_per_day){

  //0.75 kW of electricity per horsepower
  return (pump_hp*0.75)*hrs_per_day;

}
//The cost to run the plant (uses energy defined previously to determine the energy, and uses cost of electricity to find the cost)
double pump_cost (double electricity_cost, double pump_hp, double hrs_per_day){

  return energy(pump_hp, hrs_per_day)*(electricity_cost);

}

//Returns the amount you'd get out of the filter
double filter(double h2o_in, double nacl_concentrate){
  return ((h2o_in - (nacl_concentrate / 1000))-((h2o_in - (nacl_concentrate / 1000))/100));
}

//Returns how much time it will take
double time_filter(double time_per_liter, double filter_count, double h2o_in){

  return time_per_liter*h2o_in/filter_count;
}

//Returns the cost of filtering
double filter_cost(double cost_day, double hrs_per_day,double filter_time){

  return cost_day/hrs_per_day/60*filter_time;
}

//Returns the maintainance cost
double maintaining(double matinanceCost, double employeenumber, double employeeSalary, double hours_run_per_day){

  return (matinanceCost)+(employeenumber*employeeSalary*hours_run_per_day);

}

//Returns the total cost of filtering
double total_cost(double maintaining_cost_per_day, double water_cost_per_day){

  return maintaining_cost_per_day+water_cost_per_day;

}
//Also returns the total cost of filtering, but converts water cost per liter to cost per day
```

```
52. double total_cost_complex(double maintaining_cost_per_day, double
    water_cost_per_liter, double time_to_filter_one_liter, double hours_run_per_day){
53.
54.   return maintaining_cost_per_day+
    ((water_cost_per_liter*60*hours_run_per_day)/time_to_filter_one_liter);
55.
56. }
57.
58. //Returns the distillation output
59. double distill_out(double water_in){
60.   //You use about 1 L to distill 6 L of water
61.   return water_in-(water_in/7);
62. }
63.
64. //Returns the amount of time it takes to distill
65. double time_distill(double water_in){
66.   //63 minutes to distill
67.   return water_in*63;
68. }
69.
70. //Returns the distillation cost
71. double distill_cost(double water_in){
72.   //Source: https://www.quora.com/How-much-does-a-water-desalination-plant-cost
73.   return water_in*1.136;
74.
75. }
76. //The function that estimates the build cost if no initial cost is given
77. double build_cost (double cost_pump, double num_filters, double building_cost){
78.
79.   return (cost_pump*num_filters)+building_cost;
80.
81. }
82. //The waste of our plant
83. double waste(double water_in){
84.
85.   //Ours shouldn't waste any water, but this accomodates leaks and that sort of
    thing.
86.   return water_in*0.01;
87.
88. }
89. //The waste of distillation (mostly from cooling)
90. double distill_waste(double water_in){
91.
92.   return water_in/7;
93.
94. }
95.
96. //The amount of energy it would require to run a distillation plant
97. double distill_energy(double water_in){
98.
99.   //4-6 Kwh of power per L (Multi-Stage Flash Distillation, the normal distillation
    method)
100.   return (5*water_in);
101. }
102. //This function determines how long it would take to filter one liter of water
     (calculates flow rate, then converts to min/L)
```

```
103. double filter_time_base(double pump_hp, double distance_above_source, double
     friction_head){
104.
105.   return 1/(((3960*pump_hp)/(distance_above_source+friction_head))*3.7854);
106.
107. }
108.
```

```c
/*****************************************************************************

                            Run_First_Calculator.c

*****************************************************************************/

//Including Our Header file and standard input/output
#include<stdio.h>
#include "mangrove_madness.h"



//Main
int main(void){
  //The variables:
   //The PSU of the water (in PSU)
   double salt_input = 35.5;
   //The amount of water going into the system (in Liters)
   double water_input = 1000;
   //The number of filters in the plant
   double num_filters = 1.0;
   //The cost do desalinize one liter of water (in dollars)
   /* This cost includes the factors of
       -Electricity Fee (0.013 $/L)
    */
   double water_height = 10;//In feet
   //The number of hours that the plant runs per day
   double hrs_per_day = 10.0;
   //The cost to maintain the plant for one day (on average, estimated)
   double matinance_cost = 19.66/(72/hrs_per_day);
   //The number of employees working the plant
   double employees = 24;
   //The salary (per hour) of employees
   double employee_salary = 10;
   //The initial cost of the plant (default is 0, edited if not changed)
   double init_cost = 0;
   //The cost of electricity (Default: US average)
   double electricity_cost = 0.12;
   //The horsepower of the pump
   double pump_hp = 25;
   //The total head (friction of the pump)
   double pump_head = 244;
   //Cost for pumps (Default: average cost for centrifugal pumps (our recommended type))
   double cost_pump = 2500;
   //Average cost of a building (will not be used if a value is given in init_cost)
   double building_cost = 1000000;
   //Variable for when to exit the while loop
   double while_continue = 1;
   //The setting change choice
   int choice = 0;

  //Title
  printf("Mangrove Madness \n Salt Water Filtration System \n\n");


  //Prints the default settings
```

```
57.   printf("Current Settings:\n");
58.   printf(" Water Input: %lfL\n", water_input);
59.   printf(" Salinity: %lfPSU\n", salt_input);
60.   printf(" Number of Filters: %lf\n", num_filters);
61.   printf(" Pump Horsepower: %lf hp\n", pump_hp);
62.   printf(" Pump Head: %lf meters\n", pump_head);
63.   printf(" Electricity Cost: $%lf\n", electricity_cost);
64.   printf(" Cost to maintain one filter for one day: $%lf\n", matinance_cost);
65.   printf(" Number of Employees: %lf\n", employees);
66.   printf(" Employee Salary (assuming they work all the time the filters run):
    $%lf/hr\n", employee_salary);
67.   printf(" Cost to initially start the plant (The default value is 0, and will
    calculate based on an estimated building cost and the amount of filters if left at
    0): $%lf\n\n", init_cost);
68.   printf(" Height above water source: %lf feet\n", water_height);
69.
70.  //Asks the user for input to change the default settings
71.  while(while_continue == 1){
72.    printf("\nTo change settings:\n1: Water Input\n2: Salinity\n3: Number of
    Filters\n4: Pump Horsepower\n5: Pump Head\n6: Electricity Cost\n7: Cost to maintain
    one filter for one day\n8: Number of Employees\n9: Employee Salary\n10: Cost to
    initially start the plant (If the value is 0, the default is based on the number of
    filters for experimental values)\n11: Height Above Water Source\n12: Exit Settings
    Setup\nSelect a number: ");
73.    scanf("%i", &choice);
74.
75.   switch (choice){
76.   case 1:
77.     printf("Please enter the amount of water that is going into the system in liters:
    ");
78.     scanf("%le", &water_input);
79.     break;
80.   case 2:
81.     printf("Please enter the salinity of the water in PSU: ");
82.     scanf("%le", &salt_input);
83.     break;
84.   case 3:
85.     printf("Please enter the amount of filters: ");
86.     scanf("%le",&num_filters);
87.     break;
88.   case 4:
89.     printf("Please enter the pump horsepower: ");
90.     scanf("%le",&pump_hp);
91.     break;
92.   case 5:
93.     printf("Please enter the pump head: ");
94.     scanf("%le", &pump_head);
95.     break;
96.   case 6:
97.     printf("What is the electricity cost in your area: ");
98.     scanf("%le", &electricity_cost);
99.     break;
100.  case 7:
101.    printf("How much to maintain for one day? ");
102.    scanf("%le", &matinance_cost);
103.    break;
104.  case 8:
```

```
105.     printf("How many employees? ");
106.     scanf("%le", &employees);
107.     break;
108.   case 9:
109.     printf("What is an employee's salary? ");
110.     scanf("%le", &employee_salary);
111.     break;
112.   case 10:
113.     printf("What was the initial cost? ");
114.     scanf("%le", &init_cost);
115.     break;
116.   case 11:
117.     printf("Enter the height above water: ");
118.     scanf("%le", &water_height);
119.     break;
120.    case 12:
121.     printf("Calculating...\n");
122.     while_continue = 0;
123.     break;
124.   default :
125.     printf("Sorry, that is not a command number\n");
126.   }
127.  }
128.  //If no amount was given for init_cost, the cost to build the plant (roughly) is
     calculated
129.  if (init_cost == 0){
130.
131.    build_cost(cost_pump, num_filters, building_cost);
132.
133.  }
134.
135.   //Creates a new file, stats.txt
136.  FILE * stats;
137.  stats=fopen("stats.txt", "w");
138.
139.  //Prints the values to the file to be graphed in python
140.  fprintf(stats,"%le\n", time_filter(filter_time_base(pump_hp, water_height,
     pump_head), num_filters, water_input));
141.  fprintf(stats,"%le\n", filter(water_input, salt_input));
142.  fprintf(stats,"%le\n", filter_cost(pump_cost(electricity_cost, pump_hp,
     hrs_per_day), hrs_per_day,filter_time_base(pump_hp, water_height, pump_head)));
143.  fprintf(stats,"%le\n", init_cost);
144.  fprintf(stats,"%le\n", total_cost(maintaining(matinance_cost, employees,
     employee_salary, hrs_per_day), pump_cost(electricity_cost, pump_hp, hrs_per_day)));
145.  fprintf(stats,"%le\n", distill_out(water_input));
146.  fprintf(stats,"%le\n", time_distill(water_input));
147.  fprintf(stats,"%le\n", distill_cost(water_input));
148.  fprintf(stats, "%le\n", total_cost_complex(maintaining(30, 220, 20, hrs_per_day),
     distill_cost(water_input), 63, hrs_per_day));
149.  fprintf(stats, "%le\n", time_filter(filter_time_base(pump_hp, water_height,
     pump_head), num_filters, water_input));
150.  fprintf(stats, "%le\n", waste(water_input));
151.  fprintf(stats,  "%le\n", distill_waste(water_input));
152.  fprintf(stats, "%le\n", energy(pump_hp,
     hrs_per_day)/hrs_per_day/60*filter_time_base(pump_hp, water_height, pump_head));
153.  fprintf(stats, "%le\n", distill_energy(water_input));
154.  fprintf(stats, "%le\n", hrs_per_day);
```

```
155.  fprintf(stats, "%le\n", water_input);
156.  fclose(stats);
157.  printf("%le\n", filter_cost(pump_cost(electricity_cost, pump_hp, hrs_per_day),
     hrs_per_day,filter_time_base(pump_hp, water_height, pump_head)));
158.  printf("%le\n", pump_cost(electricity_cost, pump_hp, hrs_per_day));
159.
160.  return 0;
161.  }
```

```
#================================================================================
#
#                            Run_Second_Plotter.py
#
#================================================================================

#Title
print("Mangrove Madness")
print("Statistics Graphing")

#Importing the necessary plotting and displaying modules
import matplotlib.pyplot as graphs
import matplotlib.axes as plt_axis

#A function that returns the y of a linear equation
def grapheq(m, x, b):
    return ((m*x)+b)

#Opening the file from the c program in read mode
path = 'stats.txt'
stats = open(path, 'r')

#Reading each line to a dictionary
stat= {}
i = 0
for line in stats:
    i+=1
    stat[i] = line

#Defining the variables as the lines in the dictionary
stats.close()
time = stat[1]
amt = stat[2]
cost = stat[3]
init_cost = stat[4]
total_cost = stat[5]
distill_amt = stat[6]
distill_time = stat[7]
distill_cost = stat[8]
total_distill_cost = stat[9]
x_time = stat[10]
waste = float(stat[11])
distill_waste = float(stat[12])
energy = float(stat[13])
distill_energy = float(stat[14])
hrs_per_day = float(stat[15])
inp = float(stat[16])
cost = float(cost)
time = float(time)
amt = float(amt)
init_cost = float(init_cost)
total_cost = float(total_cost)
distill_amt = float(distill_amt)
distill_time = float(distill_time)
distill_cost = float(distill_cost)
total_distill_cost = float(total_distill_cost)
x_time = float(x_time)
```

```
58. init_distill = 80000000
59. time_axis = 0
60. amt_axis = 0
61.
62. #Changes the font in the title and axis font dictionaries so that words don't overlap
63. title_font = {'fontsize': 16, 'fontweight' : graphs.rcParams['axes.titleweight'],
    'verticalalignment': 'baseline', 'horizontalalignment': 'center'}
64. axis_font = {'fontsize': 12, 'fontweight' : graphs.rcParams['axes.titleweight'],
    'verticalalignment': 'baseline', 'horizontalalignment': 'center'}
65.
66. #Waiting message
67. print("The plots are loading, this may take a while...")
68.
69. #Sets the labelsize of the axis labels
70. graphs.tick_params(labelsize=18)
71.
72. #Graphs the first plot (time vs amount)
73. graphs.figure(1)
74. graphs.subplot(2,2,1)
75. graphs.title("Amount of Water Produced", fontdict = title_font)
76. graphs.axis([0,60, 0,(amt/time)*60])
77. graphs.xlabel("Time (min)", fontdict = axis_font, labelpad = 11)
78. graphs.ylabel("Amount (L)", fontdict = axis_font)
79. for i in range(int(60/5)+1):
80.     graphs.plot([0,i*5], [0,grapheq(amt/(time), i*10, 0)], 'g-')
81. for i in range(int(60/5)+1):
82.     graphs.plot([0, i*5], [0,grapheq(distill_amt/(distill_time), i*10, 0)], 'b--')
83.
84. #Graphs the second plot (input vs cost)
85. graphs.subplot(2,2,2)
86. graphs.axis([0, 100, 0, distill_cost*(100/inp)])
87. graphs.title("Cost to filter", fontdict = title_font)
88. graphs.xlabel("\nWater Input (L)", fontdict = axis_font, labelpad = 11)
89. graphs.ylabel("Cost ($)", fontdict = axis_font)
90. for i in range(int(100/10)+1):
91.     graphs.plot([0, i*10], [0, grapheq(cost, i*10, 0)], 'g-')
92. for i in range(int(100/10)+1):
93.     graphs.plot([0, i*10], [0, grapheq(distill_cost/inp, i*10, 0)], 'b--')
94.
95. #Graphs the third plot (time vs cost+cost to engineer plant)
96. graphs.subplot(2,1,2)
97. graphs.title("Total Cost", fontdict=title_font)
98. graphs.xlabel("Time (days)", fontdict = axis_font, labelpad = 14)
99. graphs.ylabel("Cost (Hundred Million $)", fontdict = axis_font, labelpad = 11)
100. for i in range(200):
101.     graphs.axis([0, i+1, 0, init_distill+(total_distill_cost*i)])
102.
103. for i in range (200):
104.     graphs.plot([0, i], [init_cost, grapheq(total_cost, i, init_cost)], 'g-')
105.
106. for i in range(int(200)):
107.     graphs.plot([0, i], [init_distill,grapheq(total_distill_cost, i, init_distill)],
    'b--')
108.
109.
110. #Graphs the fourth plot (input vs waste)
111. graphs.figure(2)
```

```
graphs.subplot(2,2,1)
graphs.title("Amount of Water Wasted", fontdict = title_font)
graphs.axis([0,100, 0,((distill_waste*(100/inp)))])
graphs.xlabel("Water Input (L)", fontdict = axis_font, labelpad = 11)
graphs.ylabel("Amount (L)", fontdict = axis_font)
for i in range(int(100/10)+1):
    graphs.plot([0,i*10], [0,grapheq(waste/(inp), i*10, 0)], 'g-')
for i in range(int(100/10)+1):
    graphs.plot([0, i*10], [0,grapheq(distill_waste/(inp), i*10, 0)], 'b--')

#Graphs the fifth plot (input vs energy)
graphs.subplot(2,1,2)
graphs.title("Amount of Electricity Used", fontdict = title_font)
graphs.axis([0,100, 0,((distill_energy*(100/inp)))])
graphs.xlabel("Water Input (L)", fontdict = axis_font, labelpad = 11)
graphs.ylabel("Electricity Used (KwH)", fontdict = axis_font)
for i in range(int(100/10)+1):
    graphs.plot([0,i*10], [0,grapheq(energy, i*10, 0)], 'g-')
for i in range(int(100/10)+1):
    graphs.plot([0, i*10], [0,grapheq(distill_energy/(inp), i*10, 0)], 'b--')


#Shows the graphs & Prints the Key
print("\n================================Key================================")
print("Green Solid Lines are Desalination via Mangrove-Mimicking Filters")
print("Blue Dashed Lines are Distillation.")
print("Figure 1: Economic Impact")
print("Figure 2: Environmental Impact")
print("===================================================================")
print("The Program Runs Until You Close Out Of The Graphs.")
graphs.show()

#Closes pyplot
graphs.close()

```