```
/* Machine project by ERMITANO, KATE JUSTINE U. [12073164] under CCPROG1 - S14A
Began program by November 24, 2020 23:51
To be submitted to Professor Danniel L. Alcantara / Professor John Alexander
Santillana
Within February 1, 2021*/

/* Accomplished all features by December 31, 2020 21:52 */

/* Overall program finished by January 30, 2021 21:42 after fixing features*/

/* This machine project has been checked by Professor Danniel L. Alcantara and
Professor John Alexander Santillana
Respective consultation dates: December 01, 2020 10:53 | January 22, 2021 14:00*/

/* Test document created on January 4, 2021 13:42 and accomplished by January 31,
2021 21:50 after making
some changes to the code to achieve the expected outcomes */


/* ------------------------------------------- */

#include<stdio.h>
#include<math.h>

/* ------------------------------------------- */

/*

DESCRIPTION: Function that makes one pizza
PARAMETERS: nPizzasMade - amount of pizzas to be added by one
RETURN VALUE: Returns nPizzasMade - a pizza made to the overall amount of pizzas

*/

int MakePizza(int nPizzasMade){

      /* Operation to add one pizza to the total amount of pizzas */
      nPizzasMade += 1;

      return nPizzasMade;
}

/*

DESCRIPTION: Function that sells a pizza
PARAMETERS: *nAmountofPizza - total amount of pizzas the player currently has
                  *fPizo - adds the pizza price to the total amount of pizo after
being sold
                  *fPizzaPrice - basis for computating the fPizo earned
                  nPizzasSold - pizza amount deducted after selling
RETURN VALUE: Returns nPizzasSold - the difference of all the pizzas made

*/

int SellPizza(int *nAmountofPizza, float *fPizo, float *fPizzaPrice, int
nPizzasSold){

      /* Operations to decrease the amount of pizza by one and add an amount to the
pizos
```

```
      based on the pizza price */
      *nAmountofPizza -= 1;
      nPizzasSold += 1;
      *fPizo += *fPizzaPrice;

      return nPizzasSold;
}

/*

DESCRIPTION: Function that hires cooks
PARAMETERS: nHiringCooks - A cook to be added to the crew one by one
RETURN VALUE: Returns nHiringCooks - the sum of all the cooks

*/

int HireCooks(int nHiringCooks){

      /* Operation that adds one cook to the total amount of cooks */
      nHiringCooks += 1;

      /* Print results */
      printf("\n| ================================ |\n\nYou hired a cook!\n");

      return nHiringCooks;
}

/*

DESCRIPTION: Function that attracts customers
PARAMETERS: nAttractingCustomer - A customer to be added one by one
RETURN VALUE: Returns nAttractingCustomer - the sum of all the customers

*/

int AttractCustomers(int nAttractingCustomer){

      /* Operation that adds one customer to the total amount of customers */
      nAttractingCustomer += 1;

      /* Print results */
      printf("\n| ================================ |\n\nYou attracted a customer!\
n");

      return nAttractingCustomer;
}

/*

DESCRIPTION: Function that computes for the new pizza price
PARAMETERS: fResearchPrice - variable to be inputted to compute for the square root
                  *fPizo - to subtract the research price from the total amount of
pizo
                  fPizzaPrice - variable to be updated by adding the square root of
the research price to the old pizza price
                  *ncounter - variable to continue to loop
                  *fOldResearchPrice - to compare the old research price with the
new input to prevent price lowering
                  *nDay - in case the player cancels the process, the day will not
```

```
be changed
RETURN VALUE: Returns fPizzaPrice - the new pizza price

*/

float ResearchPizzaQuality(float fResearchPrice, float *fPizo, float fPizzaPrice,
int *ncounter, float *fOldResearchPrice, int *nDay){

      /* Loop to make sure the user only inputs a research price not more than the
pizo they have*/

      while(*ncounter == 0){
            printf("\n| ================================ |\n");
            printf("\nNOTE: Input 0 to cancel this option\n");
            printf("\nHow much for research? ");
            scanf("%f", &fResearchPrice);

            /* Condition that will only execute the given block of code if the user
inputs
            an amount less than or equal to the amount of pizo and if the new
research price is higher than the old research price
            to prevent lowering of the price */
            if(fResearchPrice > *fOldResearchPrice && fResearchPrice <= *fPizo &&
fResearchPrice > 0 && fResearchPrice > fPizzaPrice * fPizzaPrice){
                  *fPizo -= fResearchPrice;
                  fPizzaPrice = sqrt(fResearchPrice);

                  printf("\n| ================================ |\n\nPizza price is
now %0.2f pizo \n", fPizzaPrice);

                  *ncounter += 1;
            }

            /* Condition to be executed when the input is way beyond the pizo
amount */
            else if(fResearchPrice > *fPizo){
                  printf("\n| ================================ |\n\n");
                  printf("You do not have enough pizo to do the research. Please
input a different amount. \n");

                  *ncounter = 0;

            }

            /* Condition to be executed when the input is negative */
            else if(fResearchPrice < 0){
                  printf("\n| ================================ |\n\n");
                  printf("INVALID INPUT \nTry again \n \n");

                  *ncounter = 0;
            }

            /* Condition to be executed when the input is below the old research
price */
            else if(fResearchPrice <= *fOldResearchPrice && fResearchPrice != 0){
                  printf("\n| ================================ |\n\n");
                  printf("You cannot lower your pizza price back down. \nInput a
higher price. \n \n");
```

```
                  *ncounter = 0;
            }

            /* Condition to be executed if ever the player changes their mind and
wants to cancel the process */
            else if(fResearchPrice == 0){

                  *ncounter += 1;
                  *nDay -= 1;
            }
      }

      /* Set input research price to a placeholder for comparisons */
      *fOldResearchPrice = fResearchPrice;

      /* Reset counter variable */
      *ncounter = 0;

      return fPizzaPrice;
}

/* ------------------------------------------- */

/* NIGHT PHASE */

/*

DESCRIPTION: - Function that makes pizza based on the amount of cooks present;
                  - Function that calculates the amount of paycheck every cook
demands
                        and subtracts that amount to the amount of Pizo earned;
                  - Function that eliminates cooks if ever the amount of Pizo does
not
                        meet their desired amount of paycheck;
PARAMETERS: *nCooks - adding this parameter to manipulate how many cooks stay and
resign.
                  *fPizo - fPizo is needed to subtract the paycheck all chefs
demand and to retrieve
                              the difference as the amount of pizos left.
                  nPizzas - parameter included to add the nPizzasMade to this total
variable (see parameter below).
                  nPizzasMade - parameter included to add the pizzas made to the
pizzas stored.
RETURN VALUE: Returns nPizzas - the amount of pizzas made during the night phase.

*/

int ManageCooks(int *nCooks, float *fPizo, int nPizzas, int nPizzasMade){
      /* INITIALIZATION */
      int i,
            j,
            nResignCooks = 0;

      float fPaycheck = 0,
            fPaycheckPlaceHolder,
            fPizoPlaceHolder = *fPizo,
            fConstantPizo = *fPizo,
            fPizoDifference;
```

```
      /* Condition to eliminate all chefs if ever the pizo is at zero */
      if(*fPizo <= 0){
            nResignCooks = *nCooks;
            *nCooks *= 0;
      }

      /* Condition that starts the instructions below if there are chefs hired */
      if(*nCooks > 0){

            /* Loop that calculates the possible amount of paycheck all chefs ask
*/
            for(i = 1; i <= *nCooks; i++){
                  fPaycheckPlaceHolder += i;
                  fPizoPlaceHolder -= i;
                  fPizoDifference = fConstantPizo - fPaycheckPlaceHolder;

                  /* Condition that if ever the difference between the total pizo
and paycheck are negative
                  all unpaid chefs will be eliminated */
                  if(fPizoDifference < 0){
                        *nCooks -= 1;
                        nResignCooks += 1;
                        i--;
                  }
            }

            /* Loop that subtracts the amount of paycheck from the total amount of
pizo */
            for(j = 1; j <= *nCooks; j++){
                  fPaycheck += j;
                  *fPizo -= j;
            }

            /* Make pizza after paying cooks */
            nPizzasMade += *nCooks;
      }

      /* Printing results after the above instructions have been executed */
      printf("\n| ================================ |\n\n%d pizza(s) made. \n%d
cook(s) paid with %.2f pizo. %d cook(s) resigned.\n"
      , nPizzasMade, *nCooks, fPaycheck, nResignCooks);

      /* Operation to add the amount of pizzas made by the cooks to the total
amount of pizzas*/
      nPizzas += nPizzasMade;

      return nPizzas;
}

/* ------------------------------------------- */

/*

DESCRIPTION: - Function that determines the amount of customers that arrive based
on the day;
             - Function that subtracts the pizzas bought per customer from the
total amount if pizzas;
                  - Function that adds the amount of pizo earned after selling
pizzas;
```

```
                  - Function that elimiates customers based on who does not get to
order a pizza;
PARAMETERS:  nDay - this parameter dictates which customers come to the restaurant
based on whether the customer number
                              is a multiple of the day number.
                  fPrice - parameter needed to multiply to the amount of pizzas
sold to calculate the pizos earned.
                  *nPizza - to subtract pizzas sold from the total amount of
pizzas.
                  *nCustomers - to gather all the customers and determine which of
them come based on the day, how many customers
                                          got their pizza, and how many walked out
unsatisfied to tell the other customers their experience.
                  fPizo - fPizo to add the amount earned from the sales.
                  *nPizzasSold - parameter needed to calculate the price earned
after selling a pizza.
                  *fPizoEarned - to determine how much pizo has been earned after
the customers buy pizza.
                  *nActiontoPerform - to determine which action has been selected.
RETURN VALUE: Returns fPizo - the amount earned after selling pizzas.

*/

float HandleCustomers
(int nDay, float fPrice, int *nPizza, int *nCustomers, float fPizo, int
*nPizzasSold, float *fPizoEarned, int *nActiontoPerform){

      /* INITIALIZATION */
      /* k serves as a counter for the loop */
      /* m and n are used as tags to indicate who is the first customer that has
not recieved their pizza */
      int nCustomersLeave = 0,
            k,
            m = 0,
            n = 0,
            nCustomersTotal = 0,
            nCustomerPlaceHolder = *nCustomers,
            nPizzasBought = *nPizza,
            nCustomersSatisfied = 0,
            nTotalPizzas = *nPizzasSold;

      float fPizoPlaceholder = 0;

      /* Loop to only do the instructions below if there are customers */
      if(*nCustomers > 0){
            m = *nCustomers;

            /* Loop to count customers upon arrival */
            for(k = 1; k <= *nCustomers; k++){
                  /* Condition to check whether the customer number is a factor of
the day number */
                  if(nDay%k == 0){
                        nCustomersTotal += 1;

                        /* Condition that counts how many customers received their
order */
                        if(nPizzasBought > 0){
                              nPizzasBought -= 1;
                              nTotalPizzas += 1;
```

```
                        nCustomersSatisfied += 1;
                  }

                  /* Another condition that tags which customer did not
receive a pizza
                  and starts drama on social media about the restaurant */
                  else if(nPizzasBought <= 0 && m == *nCustomers){

                        n = k;
                        m *= 0;
                  }
            }
      }

      /* Loop to eliminate customers based on the tag */
      while(*nCustomers >= n){
            *nCustomers -= 1;
      }
}

/* Calculate the amount of pizo earned by adding the pizo earned from each
customer
and the pizo earned by how many pizzas sold*/
*fPizoEarned = fPrice * nCustomersSatisfied + fPrice * *nPizzasSold;
/* Calculate the number of customers leaving by subtracting the number of
satisfied customers from the total*/
nCustomersLeave = nCustomersTotal - nCustomersSatisfied;

/* Printing results after the above instructions have been executed */

printf("\n%d customer(s) arrived. \n%d pizza(s) sold for %.2f pizo. ",
nCustomersTotal, nTotalPizzas, *fPizoEarned);

/* Condition to make the print statement appear ONLY when there is a customer
leaving */
if(nCustomersLeave > 0){
      printf("Customer %d walked out.\n", n);
}

/* Condition to fix the customer count whenever it holds a negative value*/
if(*nCustomers < 0){
      *nCustomers = nCustomerPlaceHolder;
}

/* Condition that prevents pizzas sold being doubled after selecting option 2
*/
if(*nActiontoPerform != 2){
      *nPizza -= nTotalPizzas;
      fPizo += *fPizoEarned;
}

/* Condition that prevents pizzas sold being doubled in case the player
selects option 2 */
else if(*nActiontoPerform == 2){
      nTotalPizzas -= 1;
      *fPizoEarned -= fPrice;
      *nPizza -= nTotalPizzas;
      fPizo += *fPizoEarned;
}
```

```
      return fPizo;
}

/* ------------------------------------------- */

/* MAIN PROGRAM WITH THE LEVEL INDICATOR, STATUS, AND THE ACTION MENU */
int main(){

      printf("WELCOME TO PIZZA PICKER!\n\n");

      /* Declaration of variables that are at default */

      int nDay = 0,
            nPizzas = 0,
            nCooks = 0,
            nCustomers = 0,

            /* Counters */

            nPizzasMade = 0,
            nPizzasSold = 0,
            nSkipDays = 0,
            ncounter = 0,
            nNoPizza = 0,

            nActiontoPerform;

      float fPizo = 0.00,
            fPizzaPrice = 2.00,
            fResearchPrice = 0.00,
            fOldResearchPrice = 0.00,
            fPizoEarned = 0;

      /* Do while loop to keep repeating the same instructions overtime unless the
player chooses
      an option that will break the loop (see Action 7) */
      do{
            /* LEVEL INDICATOR AND STATUS */
            nDay += 1;

            printf("\n| ================================ |\n\n");
            printf("PIZZA PICKER\n\n");
            printf("DAY %d \n", nDay);
            printf("Pizza(s): %d \n", nPizzas);
            printf("Pizo(s): %3.2f \n", fPizo);
            printf("Cook(s): %d \n", nCooks);
            printf("Customer(s): %d \n", nCustomers);
            printf("Pizza Price: %3.2f \n\n", fPizzaPrice);

            /*ACTION MENU*/
            printf("ACTION MENU: \n");
            printf("1.) Make Pizza \n");
            printf("2.) Sell Pizza \n");
            printf("3.) Hire Cook \n");
            printf("4.) Attract Customer \n");
            printf("5.) Research Pizza Quality \n");
            printf("6.) Skip No. of Days \n");
            printf("7.) End Game \n\n");
```

```
            printf("Select an action to perform: ");
            scanf("%d", &nActiontoPerform);
            printf("\n");

            /* ------------------------------------------- */

            /* Switch Case Break condition to determine which action the user
chooses to execute their
            corresponing instructions */
            switch(nActiontoPerform){
                  case 1:
                        /* Calling the MakePizza function to make a pizza */
                        nPizzasMade = MakePizza(nPizzasMade);
                  break;

                  /* ------------------------------------------- */

                  case 2:
                        /* Condition to check if there are pizzas left to be sold
*/
                        if(nPizzas > 0){
                              /* Calling the SellPizza function to sell a pizza and
gain pizo */
                              nPizzasSold = SellPizza(&nPizzas, &fPizo,
&fPizzaPrice, nPizzasSold);
                        }

                        else{
                              nNoPizza = 1;

                              printf("\n| ================================ |\n\
n");
                              printf("You cannot sell a pizza without pizzas!\n");

                              /* Subtract one day to prevent doubling */
                              nDay -= 1;
                        }
                  break;

                  /* ------------------------------------------- */

                  case 3:
                        /* Calling the HireCooks function to add a cook to the crew
*/
                        nCooks = HireCooks(nCooks);
                  break;

                  /* ------------------------------------------- */

                  case 4:
                        /* Calling the AttractCustomers function to add a customer
*/
                        nCustomers =  AttractCustomers(nCustomers);
                  break;

                  /* ------------------------------------------- */

                  case 5:
```

```
                        /* Calling the ResearchPizzaQuality function to update the
new pizza price */
                        fPizzaPrice = ResearchPizzaQuality(fResearchPrice, &fPizo,
fPizzaPrice, &ncounter, &fOldResearchPrice, &nDay);
                  break;

                  /* ------------------------------------------- */

                  case 6:
                        /* Loop that will repeat the instructions below if the user
input is invalid */
                        while(ncounter == 0){
                              /* Print and ask for an input of days */
                              printf("Skip by how many days? ");
                              scanf("%d", &nSkipDays);

                              /* Declare a counter */
                              int o;

                              /* Condition to execute the instructions below if the
player inputs a positive integer */
                              if(nSkipDays >= 0){

                                    /* Loop to keep repeating the night phase for x
number of days */
                                    for(o = 0; o < nSkipDays; o++){

                                          /* Calling the ManageCooks function to
perform its task for the night phase */
                                          nPizzas = ManageCooks(&nCooks, &fPizo,
nPizzas, nPizzasMade);

                                          /* Calling the HandleCustomers function
to perform its task for the night phase */
                                          fPizo = HandleCustomers
                                          (nDay, fPizzaPrice, &nPizzas,
&nCustomers, fPizo, &nPizzasSold, &fPizoEarned, &nActiontoPerform);

                                          nDay += 1;
                                    }

                                    nDay -= 1;
                                    ncounter += 1;
                              }

                              /* Condition to print the below statements in case
the player inputs a negative integer */
                              else if(nSkipDays < 0){
                                    printf("\n| ================================
|\n\n");
                                    printf("INVALID INPUT \nTry again \n \n");

                                    ncounter = 0;
                              }
                        }

                        /* Reset counter back to 0 */
                        ncounter = 0;
                  break;
```

```
                  /* ------------------------------------------- */

                  case 7:
                        /* Operation to multiply the day to a negative integer to
break the loop of the entire game*/
                        nDay = nDay * -1;
                  break;

                  /* ------------------------------------------- */

                  /* The option to be executed when the people inputs a number that
is out of the range (1-7)*/
                  default:
                        printf("\n| ================================= |\n\nINVALID
INPUT \nTry again \n \n");

                        /* Loop to reset the day due to an invalid input */
                        nDay -= 1;
                  }

            /* CALLING FUNCTIONS TO PERFORM THE NIGHT PHASE */
            if(nActiontoPerform >= 1 && nActiontoPerform < 7 && nActiontoPerform !=
6 && fResearchPrice <= fPizo && nNoPizza != 1){

                  /* Calling the ManageCooks function to perform its task for the
night phase */
                  nPizzas = ManageCooks(&nCooks, &fPizo, nPizzas, nPizzasMade);

                  /* Calling the HandleCustomers function to perform its task for
the night phase */
                  fPizo = HandleCustomers(nDay, fPizzaPrice, &nPizzas, &nCustomers,
fPizo, &nPizzasSold, &fPizoEarned, &nActiontoPerform);
            }

            /* Reset counters back to zero to prevent doubling*/
            nPizzasMade = 0;
            nPizzasSold = 0;
            nNoPizza = 0;

      } while(nDay > -1);

      /* Good bye message in case the player decides to quit the game */
      printf("\nThank you for playing Pizza Picker.\nHave a great day!\n");

      return 0;
}
```