```
Script started on 2023-11-28 22:58:51+00:00 [TERM="xterm-256color" TTY="/dev/pts/0" COLUM
NS="106" LINES="62"]
\[\033[01;34m\]\w\[\033[00m\]$ pwd
/home/runner/Project-7-The-Bakery-Problem-kcp3s
\[\033[01;34m\]\w\[\033[00m\]$ ls -la
total 2564
drwxr-xr-x 1 runner runner     394 Nov 28 22:58 .
drwxrwxrwx 1 runner runner     120 Nov 28 22:49 ..
-rwxr-xr-x 1 runner runner   22856 Nov 28 22:52 a.out
-rw-r--r-- 1 runner runner      17 Oct 27 20:51 .breakpoints
drwxr-xr-x 1 runner runner      12 Jan 24  2022 .cache
drwxr-x--- 1 runner runner     446 Nov 16 16:17 .ccls-cache
-rw-r--r-- 1 runner runner    4048 Nov 17 19:31 cla19.cpp
drwxr-xr-x 1 runner runner      68 Nov 25 01:46 .lesson
-rwxr-xr-x 1 runner runner 1254392 Oct 27 20:53 main
-rw-r--r-- 1 runner runner    5326 Nov 28 22:52 main.cpp
-rwxr-xr-x 1 runner runner 1255712 Oct 27 20:53 main-debug
-rw-r--r-- 1 runner runner     449 Oct 27 20:53 Makefile
-rw-r--r-- 1 runner runner    6118 Nov 17 19:56 Patel_Lab_19.log
-rw-r--r-- 1 runner runner       0 Nov 28 22:58 Patel_Project_7.log
-rw-r--r-- 1 runner runner   16246 Nov 17 20:05 Patel_Submission_Lab_19.pdf
-rw-r--r-- 1 runner runner   20259 Nov 17 20:04 Patel_Submission_Lab_19.ps
-rw-r--r-- 1 runner runner     171 Nov 28 21:30 products.dat
-rw-r--r-- 1 runner runner    1426 Dec 21  2022 .replit
-rw-r--r-- 1 runner runner     137 Nov 17 20:04 replit.nix
\[\033[01;34m\]\w\[\033[00m\]$ cat -n main.cpp
     1  #include <array>
     2  #include <fstream>
     3  #include <iomanip>
     4  #include <iostream>
     5
     6  // Constants
     7  const int kMaxProducts = 24;
     8  const int kMaxIngredients = 30;
     9
    10  // Prototypes
    11  void open_file(std::ifstream &file);
    12  void getinfo(std::ifstream &file, int ingredients[][kMaxProducts],
    13               double ingredientsprice[], int &rows, int &cols);
    14  void calculations(int ingredients[][kMaxProducts], double ingredientsprice[],
    15                    double price[], const int rows, const int cols);
    16  void display(int ingredients[][kMaxProducts], double price[],
    17               std::string productname[], const int rows, const int cols);
    18  void mostexpensive(std::string productname[], double price[], int size);
    19
    20  // COnstant global array for product names
    21  const int Maxsizeproduct = 7;
    22  std::string productname[Maxsizeproduct] = {
    23      "Donut",     "Bagel",     "White Bread", "Kaiser Roll",
    24      "King Cake", "Apple Pie", "Cherry Wafer"};
    25
    26  int main() {
    27    std::ifstream file;
    28    // using constants
    29    int rows, cols;
    30
    31    // Assigning all the arrays;
    32    int ingredients[kMaxIngredients][kMaxProducts];
    33    double ingredientsprice[kMaxProducts];
    34    double price[kMaxProducts];
    35
    36    // Using the first function
    37    open_file(file);
    38
    39    // Here we would call out functions
    40    getinfo(file, ingredients, ingredientsprice, rows, cols);
    41    // Closing the file
    42    file.close();
    43    calculations(ingredients, ingredientsprice, price, rows, cols);
    44    display(ingredients, price, productname, rows, cols);
```

```
 45    mostexpensive(productname, price, kMaxProducts);
 46
 47    return 0;
 48  }
 49  /*
 50  Purpose: Read an input file and check if the file open. If the file dont open,
 51  throw out an error using cout and keep asking until the user tells the right
 52  filename. Precondition/Input: ifstream file should have been created
 53  Postcondition/Output: output successfully opened for given file
 54      */
 55  void open_file(std::ifstream &file) {
 56    std::string filename;
 57
 58    // Asking for the file firsttime
 59    std::cout << "Enter the file name: ";
 60    std::cin >> filename;
 61    file.open(filename);
 62
 63    while (!file) {
 64      std::cout << "File open error.\n";
 65      std::cout << "Enter the file name: ";
 66      std::cin >> filename;
 67      file.open(filename);
 68    }
 69    if (file) {
 70      std::cout << "File Open Successfully." << std::endl;
 71    }
 72  }
 73
 74  /*
 75  Purpose: Read the information from the file, firstly rows and columns, then
 76  using that create a array for ingredents and ingredients price
 77  Precondition/input: Ifstream file should be open
 78  Postcondition/output: assign the values to ingredients and ingredientsprice
 79  array
 80      */
 81  void getinfo(std::ifstream &file, int ingredients[][kMaxProducts],
 82               double ingredientsprice[], int &rows, int &cols) {
 83    file >> cols >> rows;
 84
 85    // Reading ingredients and prices
 86    for (int i = 0; i < rows+1; i++) {
 87      for (int j = 0; j < cols; j++) {
 88        file >> ingredients[i][j];
 89      }
 90      file >> ingredientsprice[i];
 91    }
 92  }
 93
 94  /*
 95  Purpose: Perform calculations to determine the total price for each product
 96  Precondition/input: ingredients and ingredient price array
 97  Postcondition/output: Assign values to the price array
 98      */
 99  void calculations(int ingredients[][kMaxProducts], double ingredientsprice[],
100                    double price[], const int rows, const int cols) {
101    double total;
102    for (int i = 0; i < rows+1; i++) {
103      total = 0.0;
104      for (int j = 0; j < cols-1; j++) {
105        total += ingredients[j][i] * ingredientsprice[j];
106      }
107      price[i] = total;
108    }
109  }
110
111  /*
112  Purpose: Display the results, in the given format using, ingredients,
113  productname, and price Precondition/input: arrays for ingredients, price and
114  productname, as well as rows and cols Postcondition/output: Display the results
```

```
   115  in the formatted table
   116  */
   117  void display(int ingredients[][kMaxProducts], double price[],
   118               std::string productname[], const int rows, const int cols) {
   119
   120    // Header
   121    std::cout << "**********************************************************************"
   122                 "*******************************\n";
   123    std::cout << std::left << std::setw(2) << "Product         ";
   124    for (int i = 0; i < cols-1; i++) {
   125      std::cout << "Ing" << i + 1 << "\t";
   126    }
   127    std::cout << std::setw(9) << " Price" << std::endl;
   128
   129    std::cout << "----------------------------------------------------------------------"
   130                 "-------------------------------\n";
   131    for (int i = 0; i < rows+1; i++) {
   132      std::cout << std::setw(2) << i + 1 << std::setw(15) << productname[i];
   133      for (int j = 0; j < cols-1; j++) {
   134        std::cout << std::setw(8) << ingredients[j][i];
   135      }
   136      std::cout << "$ " << std::fixed << std::setprecision(2) << price[i]
   137                << std::endl;
   138    }
   139
   140    std::cout << "----------------------------------------------------------------------"
   141                 "-------------------------------\n";
   142  }
   143
   144  /*
   145  Purpose: Find most expensize product and give out its name.
   146  Precondition/Input: Array of product name and array of pricr and size of array.
   147  Postcondition/Output: Return the name of most expensive product
   148      */
   149  void mostexpensive(std::string productname[], double price[], int size) {
   150    double maxprice = price[0];
   151    int max = 0;
   152
   153    for (int i = 1; i < size; i++) {
   154      if (price[i] > maxprice) {
   155        maxprice = price[i];
   156        max = i;
   157      }
   158    }
   159    std::cout << std::endl;
   160
   161    std::cout << "Product " << max + 1 << ": " << productname[max]
   162              << " is the most expensive product.\n";
   163  }
\[\033[01;34m\]\w\[\033[00m\]$ g++ main.cpp -o./bakery
\[\033[01;34m\]\w\[\033[00m\]$ ./bakery
Enter the file name: products.dat
File Open Successfully.
**********************************************************************************
******
Product         Ing1    Ing2    Ing3    Ing4    Ing5    Ing6     Price
----------------------------------------------------------------------------------
------
1 Donut          10      20      50      25      0       0       $ 718.90
2 Bagel          10      5       10      25      0       0       $ 225.15
3 White Bread    11      6       12      30      0       0       $ 265.04
4 Kaiser Roll    11      6       12      30      0       0       $ 265.04
5 King Cake      5       50      40      90      0       1       $ 1085.45
6 Apple Pie      8       15      30      40      20      0       $ 909.87
7 Cherry Wafer   12      27      25      10      0       15      $ 1649.68
----------------------------------------------------------------------------------
------

Product 7: Cherry Wafer is the most expensive product.
\[\033[01;34m\]\w\[\033[00m\]$ exit
```

```
Script done on 2023-11-28 22:59:54+00:00 [COMMAND_EXIT_CODE="0"]
```