

Honors Computer Science C++

Mr. Clausen

Program 13A

Special Note: Every program from Chapter 4 to the end of the year needs to have functions!

PROBLEM 13A: Pizza_Menu (20 points)

Write a program to display all the possible pizza combinations with 4 choices for different crust types, 3 choices of a type of meat, and 5 choices of another topping. To make this program easier, the customer can only have 1 choice of crust, 1 choice of meat, and 1 choice of another topping on a pizza. This will result in 60 combinations of pizza to display on the screen. Use enumerated types for the crust type, meat type, and topping type.

Save the program as LastNameFirstNameP13A.cpp in your “S:” directory. To see a model for the menu setup for this program look at the source code for the program “Birthday.cpp” in the network directory titled: HonorsCompSciCFiles. Look in the folder, enum.

As you type all your programs this year, be sure not to type past the 80-column line in Borland C++ 5.02 for Windows. If you have any statements longer than 80 columns, press the return key to “wrap” the statement around to the next line.

1. Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Make sure to change the program name and program description in these comments, so that the program number, name, and description say what is listed above.**
2. Include `<iostream.h>` so we can use the **cout** and **cin** commands. Also include `<conio.h>` so you can use **getch()** to leave your output displayed on the screen until the user presses any key to continue. Include `<iomanip.h>` so we can use the **setw** command.
3. There are no constants to declare for this program.
4. After the constant declarations and before the function declarations is where enumerated types are to be declared. Here are the enumerations that you need to declare:

```
enum CrustType {FLAT_CRUST, THICK_CRUST, CHEESE_IN_CRUST, NEW_YORK};  
enum MeatType {HAM, PEPPERONI, SAUSAGE};  
enum ToppingType {PINEAPPLES, PEPPERS, ONIONS, MUSHROOMS, ANCHOVIES};
```

5. Don't forget to include comments for your functions, right before each of your function declarations.
6. Declare the function `Display_My_Info ()` in the function declaration section of the program. This should be a void function that doesn't take any parameters and doesn't return any values. In addition, declare these functions: **void Display_Heading; void Display_Crust(CrustType crust); void Display_Meat(MeatType meat); and void Display_Topping(ToppingType topping);**
7. Use a comment line with **equal signs** to separate all of the above from the `int main` function. For example:

```
//=====
```
8. Inside the **int main()** function, declare all of your variables right away. After that call the functions `Display_My_Info ()`, and `Display_Heading()`. You will need 3 nested for loops in the `int main` function to display all of the possible combinations of crust, meat, and topping. All of the other functions will be called from inside your nested loops. **Don't forget that value-returning functions are "called" in assignment statements, while void functions can be called just by writing the function name including any of the "actual" parameters.** Then tell the user to press any key to continue, have your `getch()` command, and your `return 0` command.
9. After the **int main()** function, have another comment line of **subtraction signs** to separate the above from your function implementation lines below. For example:

```
//-----
```
10. Implement all of the functions, separating each one from the other ones using comment lines of **subtraction signs**.
11. When you implement the function: `Display_Heading()`, use a `cout` statement to tell the user: "Here are your pizza choices: " Then, use `setw` statements to display a heading for each column in your output. Display a heading before listing each combination that describes each column of print: Crust Type, Meat Type, and Topping Type.
 To format the pizza combinations nicely use the `setw` command to set a width for each of the three columns. You also want to right justify the information in the columns. This will require that you add `#include <iomanip.h>` to your preprocessor directives and use the following command: `cout << setiosflags (ios::right) ;` and `cout << setw(20);` Remember to use switch statements for the `Display Crust`, `Meat`, and `Topping` functions. **Also remember to use explicit type conversion for each "for" loop incrementation.** For example, in the "for" loop to increment the meat type, you will increment from `HAM` to `SAUSAGE`, not the integer values that represent their positions in the

enumerated type. **If you don't do this, you will receive only half credit for your program!**

12. After the last function implementation of every program, end your program with two comment lines of equal signs. This signifies the end of your source code. For Example:

```
//=====
//=====
```

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 13A folder.