Mateo Capón Blanquer Padrón: 104258
DNI: 42496666

Hoja N°
FECHA

en ".cpp":

```
Lista<Alimento*>* BuscadorDeComidas::comidasParaCeliacos(Lista<Alimento*>* comidas, Lista<string>* ingredientesPermitidos, Lista<string>* ingredientesNoPermitidos, unsigned int caloriaMaxima){

    Lista<Alimento*>* comidasAptas = new Lista<Alimento*>;
    Alimento* actual;
    comidas->reiniciar();

    while(comidas->haySiguiente()){

        actual = comidas->siguiente();

        if((actual->obtenerCalorias() == caloriaMaxima-1)
            && (hayAlgunIngrediente(actual->obtenerIngredientes(), ingredientesPermitidos)) && (!hayAlgunIngrediente(actual->obtenerIngredientes(), ingredientesNoPermitidos))){

            agregarComida(actual, comidasAptas);
        }
    }
    return comidasAptas;
}
```

en ".h":

```
Minote: //Post: Devuelve TRUE si algun ingrediente del alimento está en el conjunto de ingredientes.
        bool hay_algun_ingrediente(Lista<string>* ingredientes_alimento,
                                   Lista<string>* conjunto_ingredientes);

//Post: Agrega el alimento a la lista de comidas.
      void agregar_comida(Alimento* actual, Lista<Alimento*>*
                          comidas);
```

en ".cpp":

```
bool Buscador_de_comidas::hay_algun_alimento(
    Lista<string>* ingredientes_comida, Lista<string>*
    conjunto_ingredientes){
     string actual;
    ingredientes_comida->reiniciar();
    conjunto_ingredientes->reiniciar();
    bool encontrado = false;
    while(!encontrado && ingredientes_comida->hay_siguiente()){
        actual = ingredientes_comida->siguiente();
        while(!encontrado && conjunto_ingredientes->hay_siguiente()){
            if(actual == conjunto_ingredientes->siguiente())
                encontrado = true;
        }
        conjunto_ingredientes->reiniciar();
    }
    return encontrado;
}
```

Mateo Copón Blanquer Padrón: 104258
DNI: 42496666

Hoja
FECHA

```
void Buscador_de_comidas::agregar_comida(Alimento*
        actual, Lista<Alimento*>* comidas){

    Lista<string>* copia = new Lista<string>;
    Lista<string>* ingredientes = actual->obtener_ingredientes();
    ingredientes->reiniciar();
    while(ingredientes->hay_siguiente()){
        copia->insertar(ingredientes->siguiente(), 1);
    }
    Alimento* nuevo = new Alimento(actual->
        obtener_nombre(), actual->obtener_calorias(), copia);
    comidas->insertar(nuevo, 1);
}
```