Declared Class Recipe:

Character string name[100] // for name of recipe

Character Ingrdts[100][100] // Ingredients

Integer num\_ingrdts // Number of Ingredients

Character instructs[100][100] // Instructions

Integer num\_instructs // Number of Instructions

Character categories[100][100] // Categories name

Integer num\_categories // Number of categories

Declared Class recipenode:

Recipe method // stores the entire recipe

recipenode\* nxt // stores the address of next node.

Declared Class CategoryNode

Character string name[100] // Category name

recipenode\* items // stores the entire recipenode class with that category

CategoryNode\* nxt // stores the address of next category Node

Declared Class IngredientNode

Character string ingredient[100] // Ingredient name

recipenode\* items // stores the entire recipenode class consisting that ingredients

IngredientNode\* nxt // stores the address of next Ingredients Node.

Initialization:

Recipe recipes[100] // Initialized an array of objects of class Recipe

Integer num\_recipes = 0 // declared a variable to count the number of recipes

CategoryNode\* categories = NULL // declared a pointer of class CategoryNode with NULL

IngredientNode\* ingrdnthashtable[100] // declared a pointer array of class IngredientNode

Declared Function hashingrdnt(character\* ingredient):

Integer hash = 0

Start for loop:

Integer i = 0

Until ingredient[i] != '\0':

hash = hash \* 31 + ingredient[i] % 100

i++

End for loop

return hash

Declared Function void addRecipe(Recipe &newRecipe):

Start if (num\_recipes < 100):

recipes[num\_recipes] = newRecipe

num\_recipes++

Start for loop:

Integer i = 0

CategoryNode\* current = categories

CategoryNode\* prev = nullptr

Start while loop:

Until ((current != nullptr) and (string compare(current->name, newRecipe.categories[i]) != 0)):

prev = current

current = current->nxt

End while

Start if (current == nullptr):

CategoryNode\* newNode = new CategoryNode

strcpy(newNode->name, newRecipe.categories[i])

newNode->items = nullptr

newNode->nxt = nullptr

Start if (prev == nullptr):

categories = newNode

else:

prev->nxt = newNode

End if

current = newNode

End if

recipenode\* newRecipeNode = new recipenode

newRecipeNode->method = newRecipe

newRecipeNode->nxt = current->items

current->items = newRecipeNode

End for

Start for loop:

Integer i = 0

Until i < newRecipe.num\_ingrdts:

Integer index = hashingrdnt(newRecipe.ingrdts[i])

IngredientNode\* newNode = new IngredientNode

String comparision(newNode->ingredient, newRecipe.ingrdts[i])

newNode->items = nullptr

newNode->nxt = ingrdnthashtable[index]

ingrdnthashtable[index] = newNode

recipenode\* newRecipeNode = new recipenode

newRecipeNode->method = newRecipe

newRecipeNode->nxt = newNode->items

newNode->items = newRecipeNode

End for

Else:

Print “Maximum number of recipes attained”

End if

Declared Function void addrecipeuser():

Start:

Recipe newRecipe

INPUT(newRecipe.name)

INPUT(newRecipe.num\_ingrdts)

Start for loop:

Integer i = 0

Until (i < newRecipe.num\_ingrdts):

INPUT(newRecipe.ingrdts[i])

i++

End for loop

INPUT(newRecipe.num\_instructs)

Start for loop:

Integer i = 0

Until (i < newRecipe.num\_instructs):

INPUT(newRecipe.instructs[i])

i++

End for loop

INPUT(newRecipe.num\_categories)

Start for loop:

Integer i = 0

Until (i < newRecipe.num\_categories):

INPUT(newRecipe.categories[i])

i++

End for loop

addRecipe(newRecipe) // Called the function to add it to LL

End function addrecipeuser

Declared Function void findrecipebyname(character\* name):

Boolean found = false

Start for loop:

Integer i = 0

Until i < num\_recipes:

Start if (string compare(recipes[i].name, name) == 0):

found = true

Print (recipes[i].name) // will print name of recipe

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_ingrdts):

Print (recipes[i].ingrdts[j])

j++

End for loop

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_instructs):

Print (recipes[i].instructs[j])

j++

End for loop

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_categories):

Print (recipes[i].categories[j])

j++

End for loop

End if

End for loop

Start if (!found):

Print(“Recipe not found”)

End if

Declared Function void searchrecipebyuser():

character string name[100]

INPUT(name)

findrecipebyname(name)

Declared Function void displayrecipebycat(character\* category):

Boolean found = false

Start for loop:

CategoryNode\* current = categories

Until current != nullptr

current = current->nxt

Start if (string compare(current->name, category) == 0):

found = true

print(category)

Start for loop:

recipenode\* recipeNode = current->items

Until (recipeNode != nullptr)

recipeNode = recipeNode->nxt

Recipe recipe = recipeNode->method

Print(recipe.name)

Start for loop:

Integer i = 0

Until(i < recipe.num\_ingrdts):

Print(recipe.ingrdts[i])

i++

End for loop

Start for loop:

Integer i = 0

Until(i < recipe.num\_instructs):

Print(recipe.instructs[i])

i++

End for loop

End if

End for loop

Start if (!found):

Print("No recipes found in category '")

End if

Declared Function void saverecipesinfile(character\* filename):

Open file(filename)

Start if(!file.is\_open()):

Print(“Error opening file”)

End if

Start for loop:

Integer i = 0

Until (i < num\_recipes):

write “Name: “ + recipes[i].name to file

write “Ingredients” to file

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_ingrdts):

write recipes[i].ingrdts[j] to file

j++

End for loop

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_instructs):

write recipes[i].instructs[j] to file

j++

End for loop

write “Categories: “

Start for loop:

Integer j = 0

Until (j < recipes[i].num\_categories):

write recipes[i].categories[j] to file

j++

End for loop

End for loop

File close // close the file

Declared Function void deleteRecipe(character\* name):

Boolean found = false

Start for loop:

Integer i = 0

Until (i < num\_recipes):

Start if (string compare (recipes[i].name, name) == 0):

found = true

Start for loop:

Integer j = i

Until j < num\_recipes - 1

j++

recipes[j] = recipes[j + 1]

End for loop

num\_recipes--

print(“Recipe deleted successfully.”)

saverecipesinfile("recipes.txt")

break

End if

End for loop

Start if (!found):

Print(“Recipe not found”)

End if

End function deleteRecipe

Declared void deleterecipebyuser():

character name[100]

INPUT(name)

deleteRecipe(name)

Declared void searchrecipesbyingredient(character\* ingredient)

Integer index= hashingrdnt(ingredient)

IngredientNode\* currentNode=ingrdnthashtable[index]

Boolean found = false

Start while

Until (currentNode!=nullptr)

recipenode\* recipeNode=currentNode->items

start while

until(recipeNode!=nullptr)

Recipe recipe=recipeNode->method;

Start for

Integer i=0

Until (i< recipe.num\_ingrdts)

i++

start if (stringcompare(recipe.ingrdts[i],ingredient)==0)

start if(!found)

found=true;

print ingredient

end if

print (recipe.name)

start for

integer j=0

until(j< recipe.num\_ingrdts)

j++

print recipe.ingrdts[j]

end for

start for

integer j=0

until(j< recipe.num\_instructs)

j++

print recipe.instructs[j]

end for

start for

integer j=0

until(j< recipe.num\_categories)

j++

print recipe.categories[j]

end for

end if

end for

recipeNode=recipeNode->nxt;

end while

currentNode=currentNode->nxt;

end while

strart if(!found)

print("No recipes found containing '")

print(ingredient)

End function searchrecipesbyingredient

Function integer main():

Recipe recipe1 = {"Aloo Mutter", {"Aloo", "mutter", "spices", "Tomato"}, 4,

{"Cut Aloo, tomato and peel off mutter as per the quantity.", "In a vessel add both vegetables and oil", "Add the spices .", "Cook until ready to serve"}, 4,

{"Indian", "Vegetable sabji"}, 2}

Recipe recipe2 = {"Chicken Curry", {"chicken","onion","tomato","coconut milk","curry powder"}, 5,

{"In a pan mix onion until soft.","Add chicken and brown on all sides.","Stir in tomato, coconut milk, and curry powder.", "Again mix until chicken is cooked through."}, 4,

{"Indian", "Curry"}, 2}

Recipe recipe3 = {"Cheese French Toast",{"Bread","Cheese","Oregano","garlic butter","Ketchup"},5,

{"Take a Bread slice and apply cheese on it.","On the other slice of bread apply Garlic butter.","Combine both the bread and toast it on a toaster","Serve with Ketchup."}, 4,{"French","Sandwich"},2}

addRecipe(recipe1)

addRecipe(recipe2)

addRecipe(recipe3)

print “Enter 1 if you want to add”

print “Enter 2 if you want to delete”

print “Enter 3 if you want to search”

print “Enter 4 if you want to display recipes by category”

print “Enter 0 if you want to exit”

integer choice

INPUT(choice)

Start while(choice):

Start switch choice:

case 1:

addrecipeuser()

break

case 2:

deleterecipebyuser()

break

case 3:

searchrecipebyuser()

break

case 4:

character category[100]

print "Enter the category to display recipes: "

INPUT(category)

displayrecipebycat(category)

break

End switch

print “Enter 1 if you want to add”

print “Enter 2 if you want to delete”

print “Enter 3 if you want to search”

print “Enter 4 if you want to display recipes by category”

print “Enter 0 if you want to exit”

INPUT(choice)

End while

Saverecipesinfile(“recipes.txt”)

Return 0

End function main