Computer Science II

CSC(17A) 47975

Professor: Dr. Mark E Lehr

Project 01

Student: Joshua McCree

Date: 10/27/2013

Project 01:

Point of this project was to create a program using the concepts from chapters 9-12 of the Gaddis book. I chose to write a program that used a lot of reading and writing to and from files and included dynamic arrays and structures for the storage and use of data.

Requirements: FIRST USE OF

* Dynamic Arrays
  + 1d Dynamic array; used line(85)
  + 2d Dynamic array; used line()
* Structures and Array Structures
  + Structure used line(46)
  + Array Structure used line (48)
* Passing and Returning Structures
  + Passing Structure used line(60)
  + Returning Structure used line (379)
* Input and Output to/from a file.
  + Reading from – fstream ios::in(“Instructions.txt”)
    - Input used line(137-148)
  + Writing to –fstream ios::out(“DinnerList.rtf”)
    - Output used line(113-119)

Pseudo Code:

Start

Include System Libraries

Include my Structures

Include function prototypes

Void Instructions(fstream, Changes)

Menu()

knwDinners(fstream)

addDinner(fstream, changes)

newDinner(string)

rmvDinner(fstream)

modify();

dlDinList(fstream)

randomDin(Dinners\*, int)

Dinners addDinList(Dinners, int, int)

Begin main body of program

Declare an initialize variables for main

Declare structures & array Structure

Do

Begin Switch

Call function menu() to display the menu and get user choice

Request user's menu choice

Choice equals case 1

Calls instructions() to Read in game instructions from a file and display to user

Choice equals case 2

Calls kwnDinners() to Read in dinners from a file, display to user

Choice equals case 3

Calls addDinner() to Add a dinner to the known dinners

Choice equals case 4

Call rmvDinner() to Remove a dinner from the known dinners

Choice equals case 5

Call dlDinner list to delete current list of dinners

And get the input for how many dinners on the new list

Choice equals case 6

Loops to fill the new dinner list

Call addDinList() to Request dinners for the new list through manual input

Choice equals case 7

Loops to fill the new dinner list with random meals

Call randomDin() to Request dinners for the list through random selection

Choice equals case 8

View the current dinner list

Choice equals 9

Export the current dinner list to a file to print/email

Choice equals anything else

Ends the program

While

choice isn't equal to 1-9

End program

void instructions(){

Pull in instructions from a file display them to user

}

short menu(){

Open file to diplay menu to user

Get input from user to make a decision

Return choice

}

void kwnDinners(){

Open file to display a current list of known dinners

}

void addDinner(){

Open file to Add a new dinner to your known dinners

Request name of recipe and needed ingredients from user

Saves input to variables adds new input and exports/updates list of known dinners

Calls newDinner() to do this step

}

void newDinner(){

Creates a new file with the added dinner name

Requests the ingredients from the user needed to make dinner

Exports those ingredients to a file that can then be accessed for new dinners

}

void rmvDinner(){

Used to remove a dinner from the known dinner list

Able to remove from specific categories

Calls modify() to remove specified dinner from the known dinner list

Uses file input and output to modify required fields

}

char modify(){

Called on by rmvDinner()

Opens file, pulls out the unwanted dinner

Then exports modified information back into the file w/o the unwanted dinner

}

void dlDinList(){

Deletes the current dinner list

Requests how many dinners will be wanted in the new dinner list

}

dinners addDinList(){

Loops until input needed has been achieved

Used to manually add dinners to the dinners list

}

void randomDin(){

Fills the dinner list with randomly chosen meals

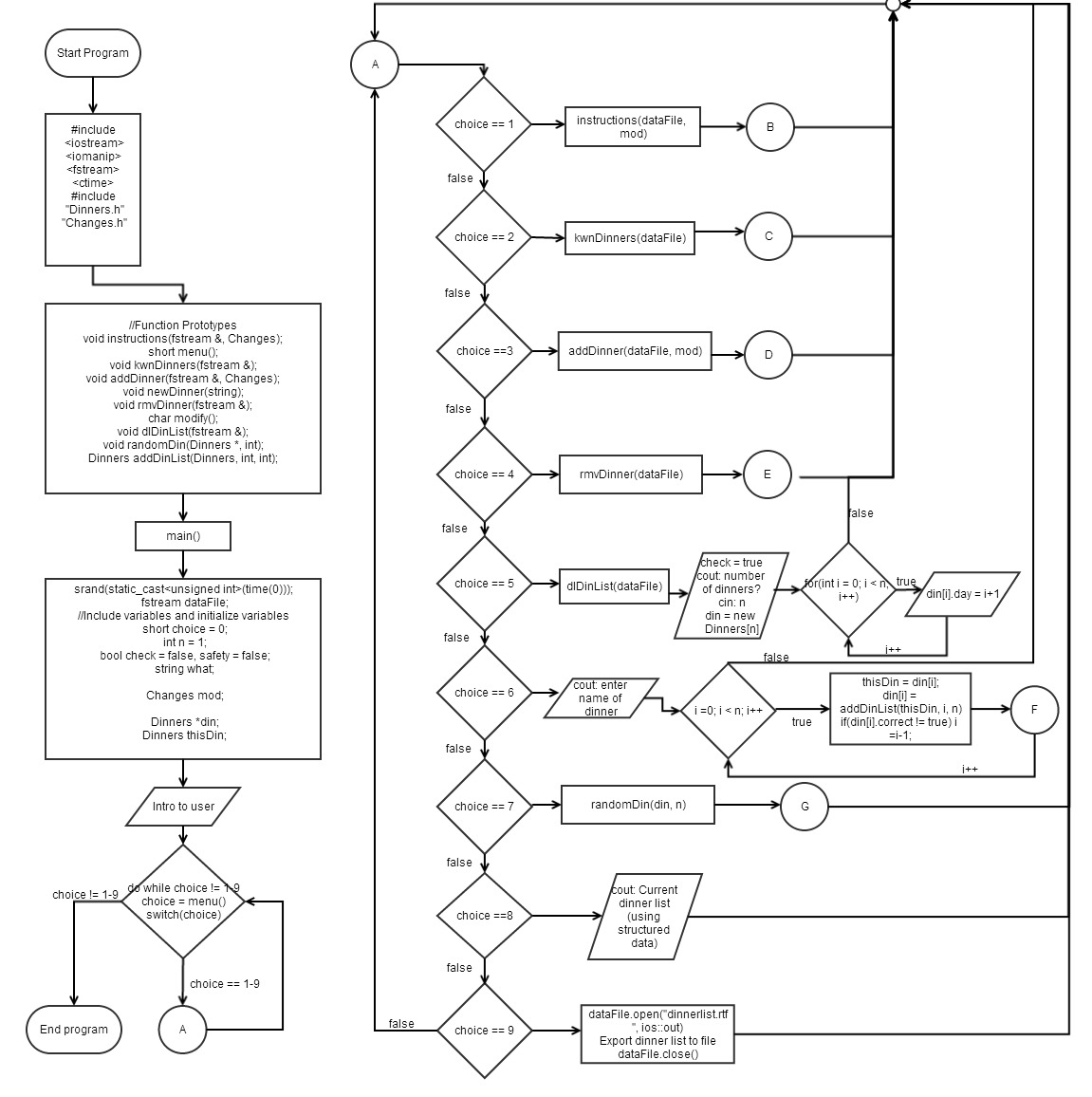
Unfortunately manually added dinners cannot be accessed under the random function

Selects random meal

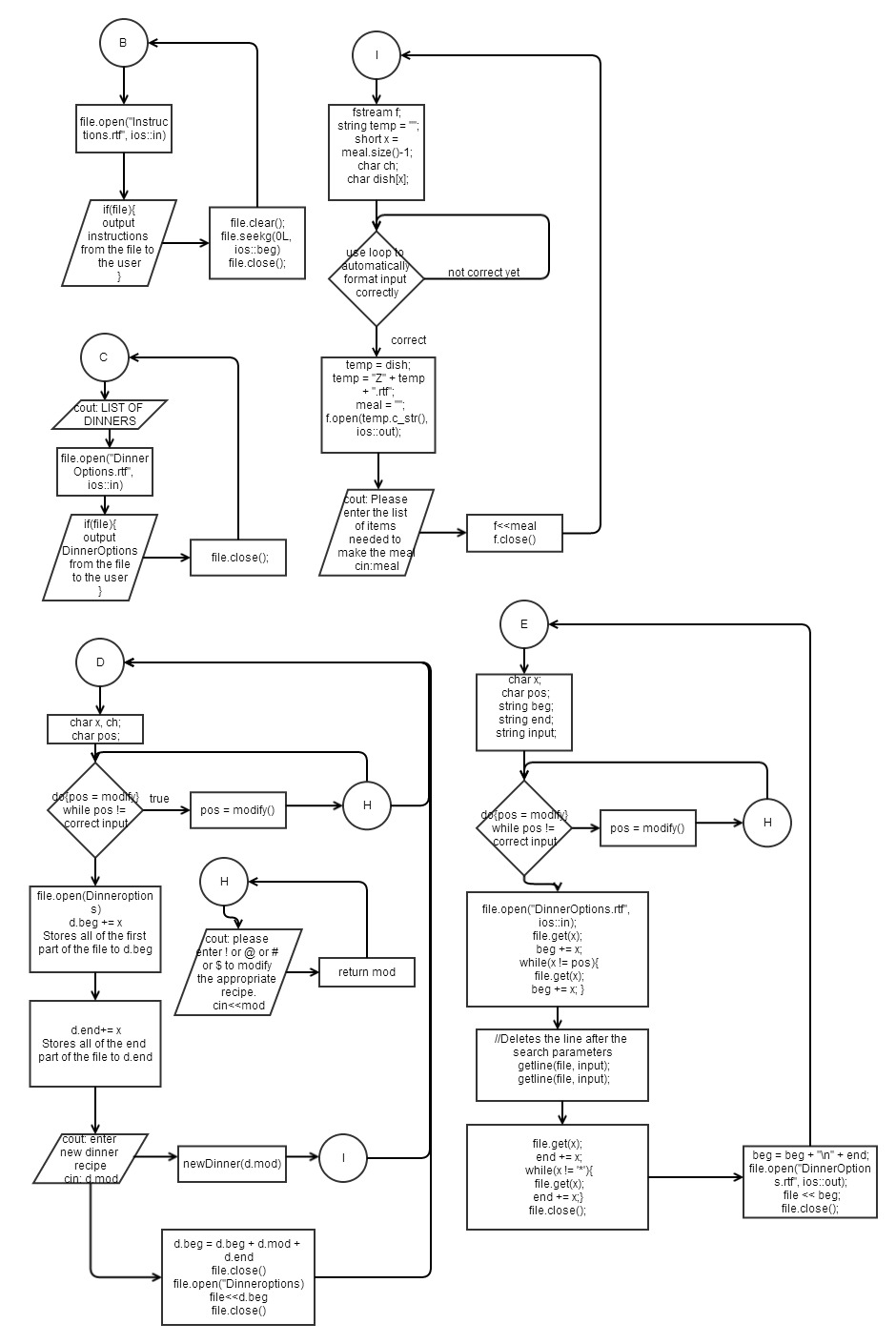
Loops until finished

}

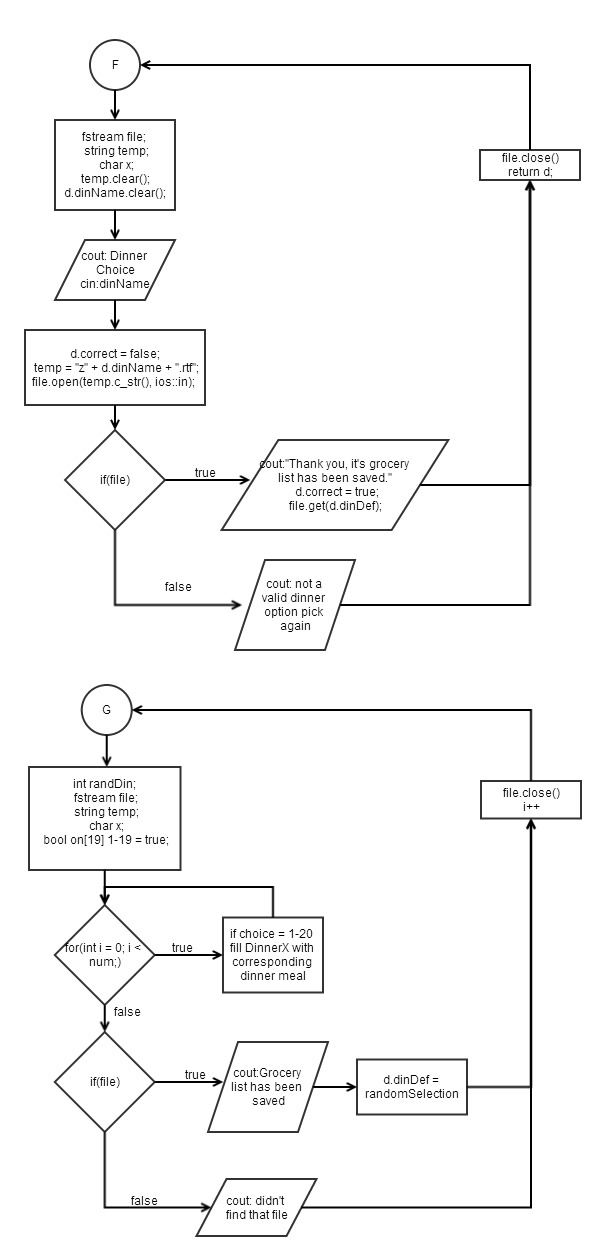
END OF CODE



Flowchart 1a



Flowchart 1b



Flowchart 1c

*PROGRAM CODE:*

/\*

\* File: main.cpp

\* Author: Joshua McCree

\*

\* Created on October 14, 2013, 7:03 PM

\*/

//Include system libraries

#include <iostream>

#include <iomanip>

#include <fstream>

#include <ctime>

using namespace std;

//Include my structures

#include "Dinners.h"

#include "Changes.h"

//Include function prototypes

void instructions(fstream &, Changes);

short menu();

void kwnDinners(fstream &);

void addDinner(fstream &, Changes);

void newDinner(string);

void rmvDinner(fstream &);

char modify();

void dlDinList(fstream &);

void randomDin(Dinners \*, int);

Dinners addDinList(Dinners, int, int);

//Begin main body of program

int main() {

srand(static\_cast<unsigned int>(time(0)));

fstream dataFile;

//Include variables and initialize variables

short choice = 0;

int n = 1;

bool check = false;

string what;

Changes mod;

Dinners \*din;

Dinners thisDin;

//Introduction to program

cout<<"Welcome to my dinner program, to begin please choose from the "

"following menu options: "<<endl<<endl;

do{

choice = menu();

if(choice > 5 && safety == false){

cout<<"You have not yet created a dinner list to view or modify"<<endl;

cout<<"So it has auto chosen this option for you to keep the program from crashing."<<endl<<endl;

choice = 5;

}

switch(choice){

case 1:

//View instructions

instructions(dataFile, mod);

break;

case 2:

//View dinners

kwnDinners(dataFile);

break;

case 3:

//Add dinner recipe

addDinner(dataFile, mod);

break;

case 4:

//Remove dinner recipe

rmvDinner(dataFile);

break;

case 5:

//Deletes the current list of dinners

dlDinList(dataFile);

//Validates that the array structure will need to be deleted

check = true;

cout<<"How many dinners would you like?"<<endl;

cin>>n;

//Creates a dynamic array of structures that will hold the new dinners

din = new Dinners[n];

for(int i = 0; i < n; i++){

din[i].day = i+1;

}

break;

case 6:

cout<<"Please enter the ("<<n<<") name(s) of a known dinner(s) you wish to add"<<endl;

cin.ignore();

for(int i = 0; i < n; i++){

thisDin = din[i];

din[i] = addDinList(thisDin, i, n);

if(din[i].correct != true)i=i-1;

}

break;

case 7:

randomDin(din, n);

break;

case 8:

//View current dinner list

for(int i = 0; i < n; i++){

cout<<"Dinner ("<<din[i].day<<"): "<<din[i].dinName<<endl;

cout<<din[i].dinDef<<endl;

}

break;

case 9:

dataFile.open("DinnerList.rtf", ios::out);

for(int i = 0; i < n; i++){

dataFile <<"Dinner (";

dataFile <<din[i].day;

dataFile <<"): ";

dataFile <<din[i].dinName;

dataFile <<"\n";

dataFile << din[i].dinDef;

dataFile <<"\n";

}

dataFile.close();

break;

default:

cout<<"You chose to exit the program..."<<endl;

choice = 10;

break;

}

} while(choice != 10);

if(check == true) delete []din;

dataFile.close();

return 0;

}

void instructions(fstream &file, Changes d){

//Opens the instruction file

file.open("Instructions.rtf", ios :: in);

//Reads in each line from the file then outputs to the user

if(file){

while(file){

getline(file, d.mod);

cout<<d.mod<<endl;

}

file.clear();

file.seekg(0L, ios :: beg);

file.close();

}

}

//Begins the menu function

short menu(){

//Declare and initialize variables

fstream file;

short c = 0;

string words;

//Opens the menu file

file.open("Menu.txt", ios :: in);

//Reads in each line from the file then outputs to the user

if(file){

while(file){

getline(file, words);

cout<<words<<endl;

}

file.clear();

file.seekg(0L, ios :: beg);

}

file.close();

//Gets the desired option from the user

cout<<"Choice: ";

cin>>c;

//Returns the users option back to the main body

return c;

}

void kwnDinners(fstream &file){

char letter;

file.open("DinnerOptions.rtf", ios :: in);

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LIST OF DINNERS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

//Stores the file input up to the insert line.

if(file){

file.get(letter);

cout<<letter;

while(letter != '\*'){

file.get(letter);

cout<<letter;

}

}

else cout<<"That file doesn't exist."<<endl;

file.close();

}

void addDinner(fstream &file, Changes d){

char x, ch;

char pos;

do{

pos = modify();

}while(pos != '!' && pos != '@' && pos !='#' && pos != '$');

file.open("DinnerOptions.rtf", ios::in);

//Stores the file input up to the insert line.

file.get(x);

d.beg += x;

while(x != pos){

file.get(x);

d.beg += x;

}

//Stores the file input after the insert line.

file.get(x);

d.end += x;

while(x != '\*')

{

file.get(x);

d.end += x;

}

//Gets the input from the user that is going to be added to the file.

cout<<"Type the name of your new dinner recipe followed by a \*"<<endl;

cout<<"and once you are done, press [ENTER] to continue."<<endl;

// Gets the name of the new dinner recipe

cin.get(ch);

while (ch != '\*') {

d.mod += ch;

cin.get(ch);

}

newDinner(d.mod);

d.beg = d.beg + d.mod + d.end;

file.close();

//Rewrites the data file with the modified changes

file.open("DinnerOptions.rtf", ios::out);

file << d.beg;

file.close();

}

//Runs a function to create the grocery list for the new dinner recipe

void newDinner(string meal){

//Declare and initialize needed variables for this function

fstream f;

string temp = "";

short x = meal.size()-1;

char ch;

char dish[x];

//Removes the enter key stroke from the name of the meal.

for(int i = 0; i < x; i++){

dish[i] = meal[i+1];

}

temp = dish;

temp = "Z" + temp + ".rtf";

meal = "";

f.open(temp.c\_str(), ios::out);

cout<<"\nPlease enter the list of items you will need to make this dish."<<endl;

cout<<"You may finish your input at anytime by using the \* key and pressing [ENTER]"<<endl;

cin.get(ch);

while (ch != '\*') {

meal += ch;

cin.get(ch);

}

meal += ch;

f<<meal;

f.close();

}

void rmvDinner(fstream &file){

char x;

char pos;

string beg;

string end;

string input;

do{

pos = modify();

}while(pos != '!' && pos != '@' && pos !='#' && pos != '$');

file.open("DinnerOptions.rtf", ios::in);

//Stores the file input up to the insert line.

file.get(x);

beg += x;

while(x != pos){

file.get(x);

beg += x;

}

//Deletes the line after the search parameters

getline(file, input);

getline(file, input);

//Stores the file input after the insert line.

file.get(x);

end += x;

while(x != '\*')

{

file.get(x);

end += x;

}

file.close();

beg = beg + "\n" + end;

file.open("DinnerOptions.rtf", ios::out);

file << beg;

file.close();

}

//Begins the modify function

char modify(){

//Declare variables

char mod;

//Requests designated input from user to determine which catagory to alter

cout<<"Would you like to modify a recipe under: "<<endl;

cout<<"(!) Chicken"<<endl;

cout<<"(@) Beef"<<endl;

cout<<"(#) Pork/Ham"<<endl;

cout<<"($) Sausage"<<endl;

cout<<"Please enter the corresponding symbol: "<<endl;

cin>>mod;

return mod;

}

void dlDinList(fstream &file){

file.open("DinnerList.rtf", ios::out);

file << " ";

file.close();

}

Dinners addDinList(Dinners d, int pos, int numDin){

fstream file;

string temp;

char x;

temp.clear();

d.dinName.clear();

cout<<"Dinner choice ("<<pos+1<<"): ";

getline(cin, d.dinName);

d.correct = false;

temp = "z" + d.dinName + ".rtf";

file.open(temp.c\_str(), ios::in);

if(file){

cout<<"Thank you, it's grocery list has been saved."<<endl<<endl;

d.correct = true;

file.get(x);

while(x != '\*')

{

d.dinDef += x;

file.get(x);

}

}

else{

cout<<"That isn't a valid dinner option. Please try naming a known dinner"<<endl;

}

file.close();

return d;

}

void randomDin(Dinners \*d, int num){

int randDin;

fstream file;

string temp;

char x;

bool failed = false, first = true;

bool on[19];

if(first == true){

for(int i = 0; i < 20; i++){

on[i] = true;

}

first = false;

}

for(int i = 0; i < num; ){

randDin = rand()%20;

if(randDin == 1 && on[0] == true){

d[i].dinName = "Chicken Rolls";

on[0] = false;

}

else if(randDin == 2 && on[1] == true){

d[i].dinName = "Enchiladas";

on[1] = false;

}

else if(randDin == 3 && on[2] == true){

d[i].dinName = "Baked Potatoes";

on[2] = false;

}

else if(randDin == 4 && on[3] == true){

d[i].dinName = "BBQ Pulled Pork";

on[3] = false;

}

else if(randDin == 5 && on[4] == true){

d[i].dinName = "Beef and Broccoli";

on[4] = false;

}

else if(randDin == 6 && on[5] == true){

d[i].dinName = "Burritos";

on[5] = false;

}

else if(randDin == 7 && on[6] == true){

d[i].dinName = "Cordon Bleu";

on[6] = false;

}

else if(randDin == 8 && on[7] == true){

d[i].dinName = "Curry";

on[7] = false;

}

else if(randDin == 9 && on[8] == true){

d[i].dinName = "Ham Fried Rice";

on[8] = false;

}

else if(randDin == 10 && on[9] == true){

d[i].dinName = "Hamburgers";

on[9] = false;

}

else if(randDin == 11 && on[10] == true){

d[i].dinName = "Lasagna";

on[10] = false;

}

else if(randDin == 12 && on[11] == true){

d[i].dinName = "Navaho Tacos";

on[11] = false;

}

else if(randDin == 13 && on[12] == true){

d[i].dinName = "Pork Chops";

on[12] = false;

}

else if(randDin == 14 && on[13] == true){

d[i].dinName = "Pot Pie";

on[13] = false;

}

else if(randDin == 15 && on[14] == true){

d[i].dinName = "Potato Soup";

on[14] = false;

}

else if(randDin == 16 && on[15] == true){

d[i].dinName = "Roast";

on[15] = false;

}

else if(randDin == 17 && on[16] == true){

d[i].dinName = "Spaghetti";

on[16] = false;

}

else if(randDin == 18 && on[17] == true){

d[i].dinName = "Stew";

on[17] = false;

}

else if(randDin == 19 && on[18] == true){

d[i].dinName = "Stir Fry";

on[18] = false;

}

else if(randDin == 20 && on[19] == true){

d[i].dinName = "Stroganoff";

on[19] = false;

}

else {

continue;

}

temp = "z" + d[i].dinName + ".rtf";

file.open(temp.c\_str(), ios::in);

if(file){

cout<<"Thank you, it's grocery list has been saved."<<endl<<endl;

file.get(x);

while(x != '\*')

{

d[i].dinDef += x;

file.get(x);

}

}

else cout<<"Didn't find that file: "<<temp<<endl;

file.close();

i++;

}

}