CompSci 2S03 – Assignment 1

Name: Jatin Chowdhary

Student #: 400033011

Mac ID: Chowdhaj

Date: September 30th, 2018

**1. One large chemical company pays its salespeople on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5000 worth of chemicals in a week receives $200 plus 9% of $5000, or a total of $650. Develop a program that will input each salesperson’s gross sales for last week and will calculate and display that salesperson’s earnings. Process one salesperson's figures at a time.**

**A) Pseudocode**

Print a divider

Print name of program to screen

Print a divider

Print brief description of program to screen

Print a divider

Declare a variable of type char

Prompt user to press Enter to begin

Store input into the variable of type char

Declare variable of type double

Start do while loop:

Print divider

Print prompt that tells user what to do

Print divider

Print prompt that tells user how to exit program

Print divider

Prompt user to enter weekly gross sales and make an alert sound

Store input into the variable of type double

If input does not equal -1.0:

Print divider

Print weekly earnings to screen

Print divider

Else:

Print divider

Print a good bye message to screen and make an alert sound

Print divider

Repeat this loop as long as the user does not enter -1.0 into the prompt. End

**B) C Code**

// Author: Jatin Chowdhary Chowdhaj

#include <stdio.h>

#include <stdlib.h>

int main() {

printf("\n##############################\n"); // print divider

printf("# Sales Comission Calculator #\n"); // print name of program

printf("##############################\n"); // print divider

printf("# This program will compute #\n"); // print description of program

printf("# weekly sales using the #\n"); // print description of program

printf("# following formula: #\n"); // print description of program

printf("# $200 + 9%% of weekly sales #\n"); // print formula used to calculate sales

printf("##############################\n\n"); // print divider

char prompt; // declare variable of type char and call it prompt

printf("Press Enter To Begin\n"); // tell user to press enter to begin

scanf(" %c", &prompt); // store input (Enter) from user into prompt

double sales; // declare variable of type double and call it sales

do { // start do while loop

printf("-------------------------------\n"); // print divider

printf("Enter Weekly Sales (In Dollars)\n"); // tell user to enter weekly sales

printf("-------------------------------\n"); // print divider

printf("Enter -1 To End Program\n"); // tell user to enter -1 to end program, whenever

printf("-------------------------------\n"); // print divider

printf("Weekly Gross Sales --> $\a"); // tell user to enter weekly gross sales here (the ‘\a’ makes an alert sound)

scanf(" %lf", &sales); // store input from user into sales

if (sales != -1.0) { // if sales is not -1.0, then do the following:

printf("-------------------------------\n"); // print divider

printf("Weekly Earnings: $%.2lf\n", (200 + sales \* 0.09)); // print weekly earnings to screen by applying the formula

printf("-------------------------------\n\n"); // print divider

} else { // if sales is -1.0, then:

printf("-------------------------------\n"); // print divider

printf("Thank You! Have A Good Day!\n\a"); // print a thank you message to screen

printf("-------------------------------\n\n"); // print divider

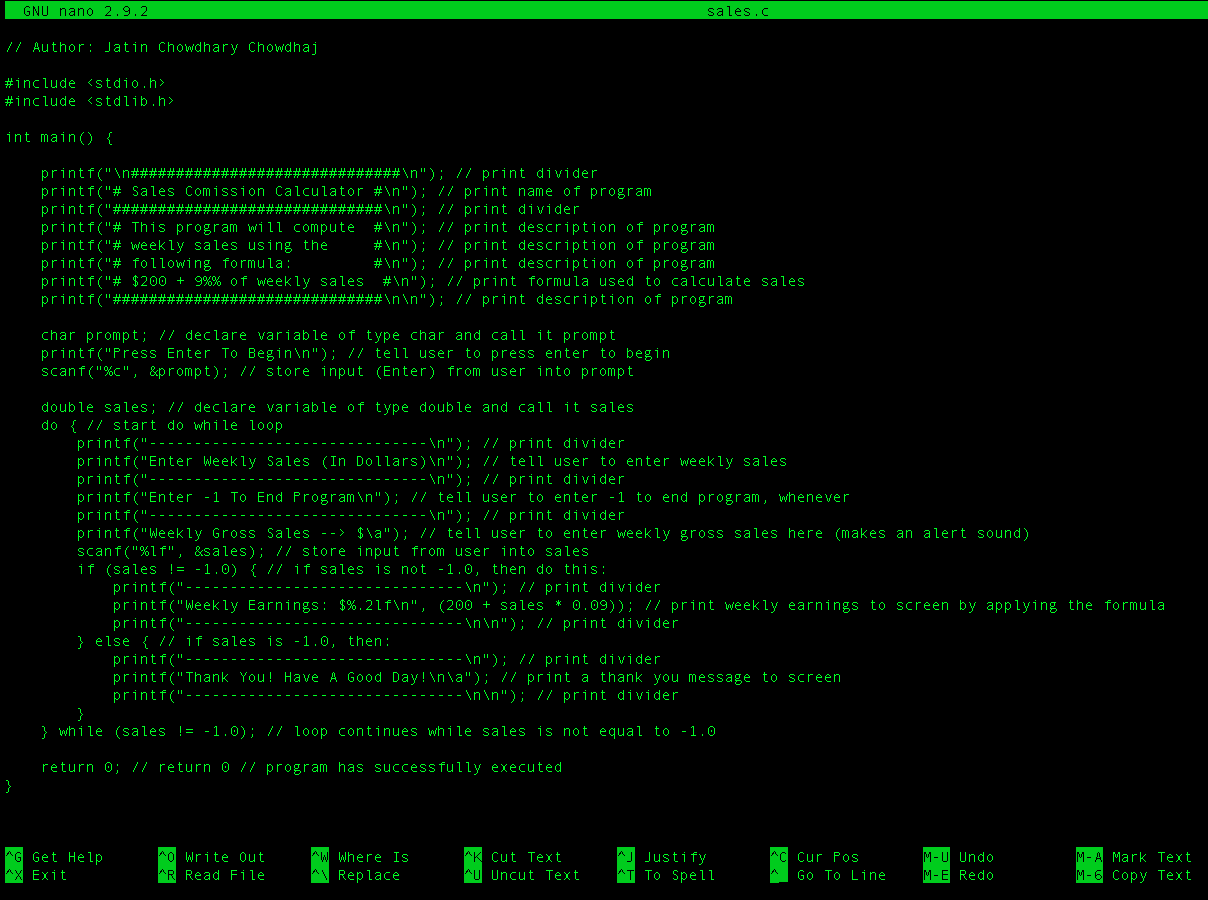
}

} while (sales != -1.0); // loop continues while sales is not equal to -1.0

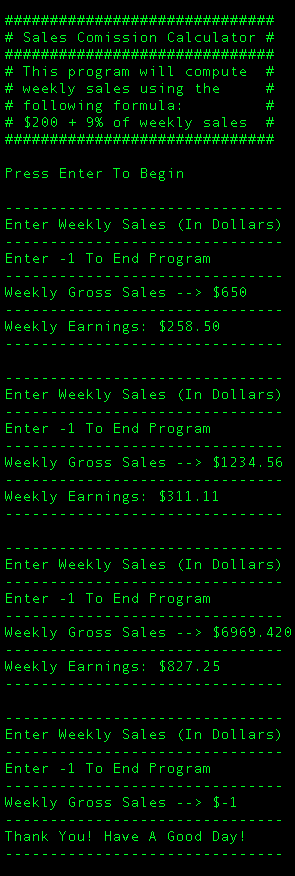
return 0; // return 0 // program has successfully executed

}

**C) Screenshot Of Code**



**D) Screenshot Of Execution**

****

**2. *(Find the Largest Number)* The process of finding the largest number (i.e., the maximum of a group of numbers) is used frequently in computer applications. For example, a program that determines the winner of a sales contest would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a pseudo code program and then a program that inputs a series of 10 non-negative numbers and determines and prints the largest of the numbers**

**A) Pseudocode**

Print divider

Print name of program

Print divider

Print divider

Print instructions

Print divider

Declare two variables of type float

Declare an int (counter) and initialize it to 1

While Loop (While counter is less than or equal to 10):

If counter does not equal ten then print 0

Print counter to screen and increment it by 1, and prompt user for input

Accept input from user and store it into a variable of type float

If counter is equal to 2 then store input into a variable of type float

If input from user is greater than other float variable, other float

variable gets the value of input from user

Loop continues until counter hits 11; then it’ll exit out of the loop

Print divider

Print largest number to screen

Print divider

End

**B) C Code**

// Author: Jatin Chowdhary Chowdhaj

#include <stdio.h>

#include <stdlib.h>

int main() {

printf("\n##################\n"); // print divider

printf("# Largest Number #\n"); // print name of program to screen

printf("##################\n\n"); // print divider

printf("------------------\n"); // print divider

printf("\aPlease Input Ten\nNon-Negative #'s\n"); // prints instructions to screen and makes an alert sound

printf("------------------\n"); // prints divider

int counter = 1; // declares and initializes a variable of type int, called counter, and assigns a value of 1

float input; // declares a variable of type float called input

float max; // declares a variable of type float called max

while (counter <= 10) { // while counter is less than or equal to 10

if (counter != 10) printf("0"); // if counter is not 10 then print 0

printf("%d. ", counter++); // print counter to screen followed by a period, then increment counter by 1

scanf(" %f", &input); // prompt user for input and store that into the variable input

if (counter == 2) max = input; // if counter is 2 then max gets input (assign value of input to max)

if (input > max) { // if input is greater than max then...

max = input; // ... max gets input (assign input to max)

}

}

printf("------------------\n\a"); // print divider and make an alert sound

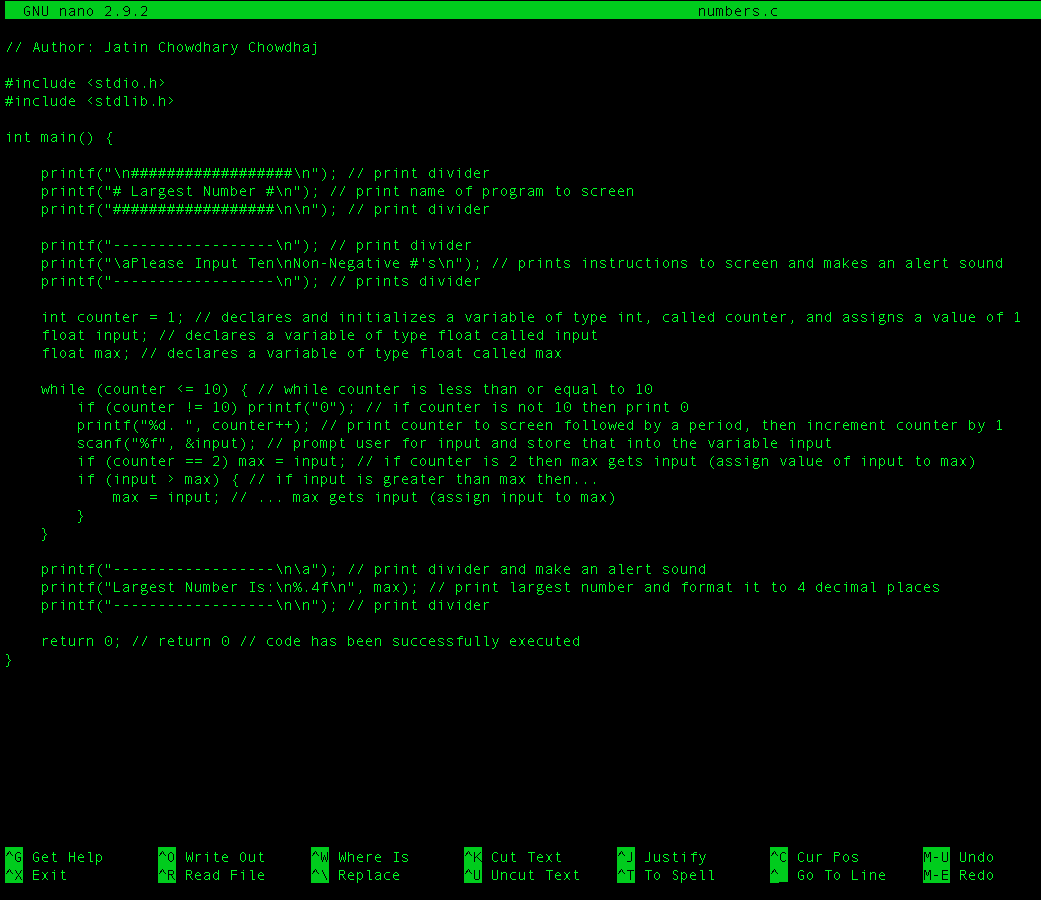
printf("Largest Number Is:\n%.4f\n", max); // print largest number and format it to 4 decimal places

printf("------------------\n\n"); // print divider

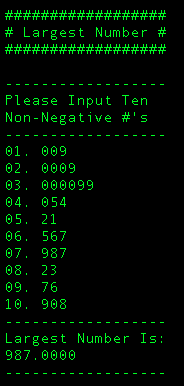
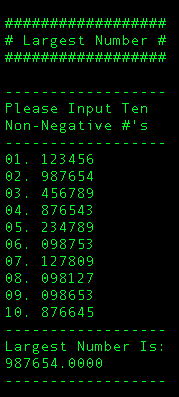
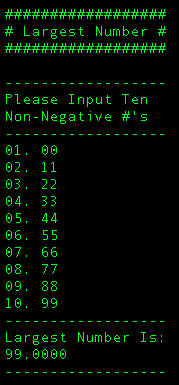
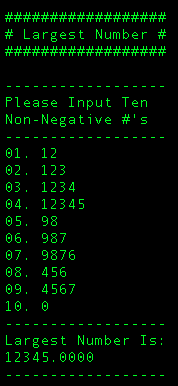
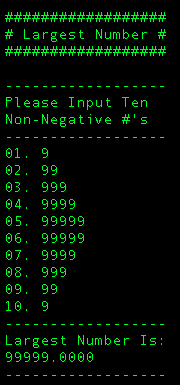
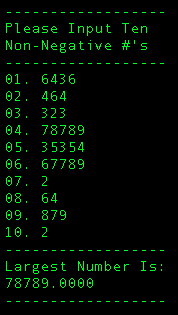
return 0; // return 0 // code has been successfully executed

}

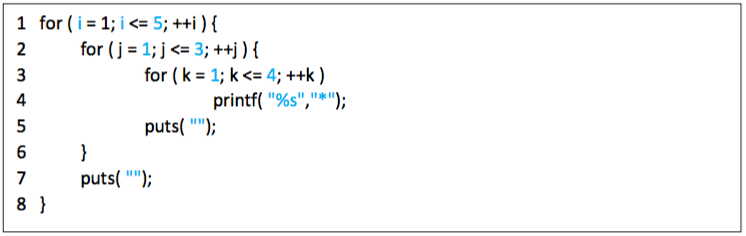
**C) Screenshot Of Code**

****

**D) Screenshot Of Execution**



3. What does the following program segment do?



**A) Pseudocode**

For Loop:

Initialize: Set i to 1

Condition: i is less than or equal to 5

Increment: Add one to i on each iteration

For Loop:

Initialize: Set j to 1

Condition: j is less than or equal to 3

Increment: Add one to j on each iteration

For Loop:

Initialize: Set k to 1

Condition: k is less than or equal to 4

Increment: Add one to k on each iteration

Print an asterisk to the screen

Print a new line (\n) using puts

Print a new line (\n) using puts

**B) C Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

int i, j, k; // initialize int i, j, k

for (i = 1; i <= 5; ++i) { // for loop that will iterate 5 times

for (j = 1; j <= 3; ++j) { // for loop that will iterate 3 times

for (k = 1; k <= 4; ++k) // for loop that will iterate 4 times

printf("%s","\*"); // prints an asterisk to the screen

puts(""); // prints a newline // similar to printf("\n")

}

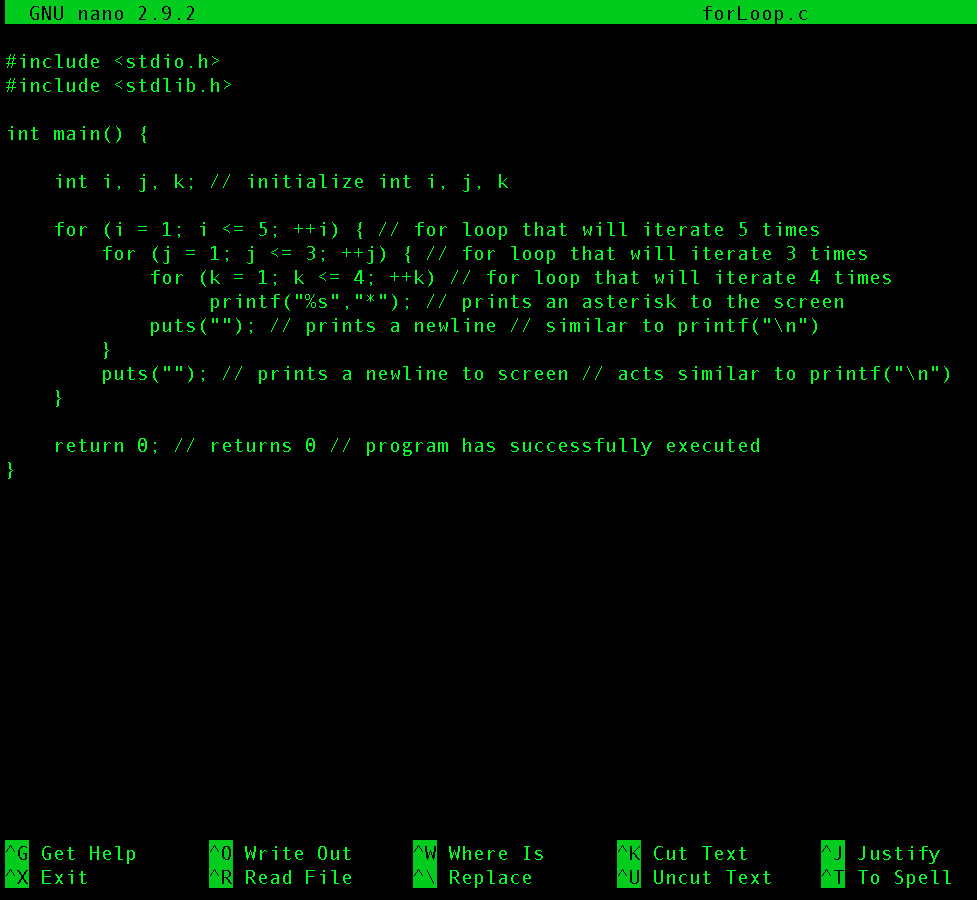
puts(""); // prints a newline to screen // acts similar to printf("\n")

}

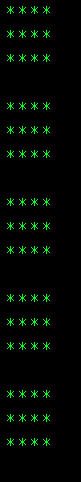
return 0; // returns 0 // program has successfully executed

}

**C) Screenshot Of Code**



**D) Screenshot Of Execution**



Answer:

The code segment (above) prints four by three rows of asterisks, five times, to the screen. Each block/rectangle of asterisks is followed (separated) by a newline. Refer to the image above to see the output of this code segment.

In this code segment, there are 3 for loops. Each for loop is responsible for a different part of the output. The first for loop is responsible for the 5 rectangles of asterisks. It also puts a space between each rectangle, making it appear as if there are 5 rectangles. The second for loop is responsible for the height of the rectangle, which is 3. Since the condition in the second for loop is j <= 3, the height of the rectangles are also 3. Also, this for loop adds a newline at the end of each iteration. This is important because it gives each rectangle (of asterisks) its shape. The third for loop is responsible for the length of the rectangle. The length of each asterisk rectangle is 4, and the condition is k <= 4.

The ‘puts(“”);’ statements are important, because without them, the asterisks would never form the shape of a rectangle. If you remove both of them, the output would be a long line of asterisks.

Also, this code segment is missing initial variable declarations for i, j, and k. Without these statements, the code will not run. These statements must be added, and then the whole thing needs to be placed inside C boilerplate code.