

COMPSCI 2S03: Assignment #1

Shawn Esseltine | 400145888

Question 1

(Sales Commission Calculator) 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.

Pseudo Code:

While user has more input (hasn't entered -1):

 Prompt the user for input

 If the input isn't -1:

 Calculate salary

 Print salary

Code:

```
1 //Author: Shawn Esseltine 400145888
2 //Question 1
3
4 //Import all required libraries
5 #include <stdio.h>
6
7 int main(void){
8     //Initialize required variables
9     unsigned int flag = 1;
10    double sales = 0, commission = 0;
11
12    //Create input loop
13    do{
14        //Prompt the user for input
15        printf("%s", "Enter sales in dollars (-1 to end): ");
16        scanf("%lf", &sales);
17        //Determine if user intends to finish program
18        if(sales == -1){
19            //Set flag to false (0)
20            flag--;
21        }else{
22            //Calculate the commission to be paid
23            commission = 0.09 * sales + 200;
24            //Print the employees salary
25            printf("%s%.2f\n", "Salary is: ", commission);
26        }
27    }while(flag);
28 }
29
```

```
//Author: Shawn Esseltine 400145888
//Question 1

//Import all required libraries
#include <stdio.h>

int main(void){
    //Initialize required variables
    unsigned int flag = 1;
    double sales = 0, commission = 0;

    //Create input loop
    do{
        //Prompt the user for input
        printf("%s", "Enter sales in dollars (-1 to end): ");
        scanf("%lf", &sales);
        //Determine if user intends to finish program
        if(sales == -1){
            //Set flag to false (0)
            flag--;
        }else{
            //Calculate the commission to be paid
            commission = 0.09 * sales + 200;
            //Print the employees salary
            printf("%s%.2f\n", "Salary is: ", commission);
        }
    }while(flag);
}
```

Execution:

```
Shawns-MacBook-Pro:~ shawn$ ./q1
Enter sales in dollars (-1 to end): 5000.00
Salary is: 650.00
Enter sales in dollars (-1 to end): 1234.56
Salary is: 311.11
Enter sales in dollars (-1 to end): -1
Shawns-MacBook-Pro:~ shawn$
```

Question 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.

Pseudo Code:

Loop 10 times:

 Prompt the user for input

 If inputted number is larger than the current largest number:

 Largest number becomes inputted number

Print the largest number

Code:

```
1 //Author: Shawn Esseltine 400145888
2 //Question 2
3
4 //Import all required libraries
5 #include <stdio.h>
6
7 int main(void){
8     //Initialize required variables
9     unsigned int number = 0, largest = 0;
10
11     //Create input loop
12     for(unsigned int counter = 1; counter <= 10; counter++){
13         //Prompt the user for input
14         printf("%s", "Input number of sales: ");
15         scanf("%u", &number);
16
17         //Check if input is greater than the previously largest
18         if(number > largest){
19             //Update the largest to be the inputted number
20             largest = number;
21         }
22     }
23     //Display the largest number of sales
24     printf("%s %u\n", "Largest number of sales: ", largest);
25 }
26
```

```
//Author: Shawn Esseltine 400145888
//Question 2

//Import all required libraries
#include <stdio.h>

int main(void){
    //Initialize required variables
    unsigned int number = 0, largest = 0;

    //Create input loop
    for(unsigned int counter = 1; counter <= 10; counter++){
        //Prompt the user for input
        printf("%s", "Input number of sales: ");
        scanf("%u", &number);

        //Check if input is greater than the previously largest
        if(number > largest){
            //Update the largest to be the inputted number
            largest = number;
        }
    }
    //Display the largest number of sales
    printf("%s %u\n", "Largest number of sales: ", largest);
}
```

Execution:

```
Shawns-MacBook-Pro:Al shawn$ ./q2
Input number of sales: 1
Input number of sales: 2
Input number of sales: 3
Input number of sales: 4
Input number of sales: 5
Input number of sales: 6
Input number of sales: 7
Input number of sales: 8
Input number of sales: 9
Input number of sales: 10
Largest number of sales: 10
Shawns-MacBook-Pro:Al shawn$
```

```
Shawns-MacBook-Pro:Al shawn$ ./q2
Input number of sales: 5
Input number of sales: 50
Input number of sales: 22
Input number of sales: 90
Input number of sales: 82
Input number of sales: 1
Input number of sales: 70
Input number of sales: 89
Input number of sales: 19
Input number of sales: 90
Largest number of sales: 90
Shawns-MacBook-Pro:Al shawn$
```

Question 3

What does the following program segment do?

```
1 for ( i = 1; i <= 5; ++i ) {
2     for ( j = 1; j <= 3; ++j ) {
3         for ( k = 1; k <= 4; ++k )
4             printf( "%s", "*" );
5         puts( "" );
6     }
7     puts( "" );
8 }
```

Pseudo Code:

Loop 5 times:

 Loop 3 times:

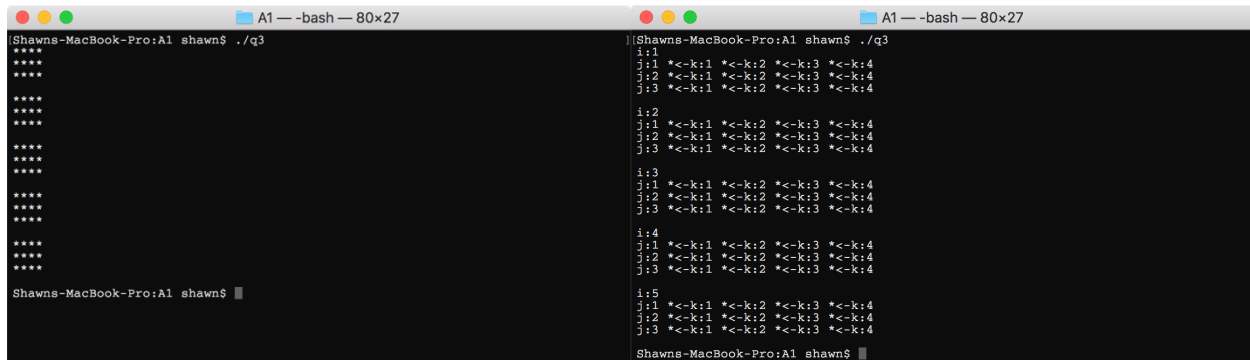
 Loop 4 times:

 Display “*”

 Break line

 Break line

Execution:



```
Shawns-MacBook-Pro:~ shawn$ ./q3
****
****
****
****
****

Shawns-MacBook-Pro:~ shawn$

Shawns-MacBook-Pro:~ shawn$ ./q3
i:1
j:1 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:2 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:3 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
i:2
j:1 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:2 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:3 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
i:3
j:1 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:2 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:3 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
i:4
j:1 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:2 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:3 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
i:5
j:1 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:2 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
j:3 *-<-k:1 *-<-k:2 *-<-k:3 *-<-k:4
Shawns-MacBook-Pro:~ shawn$
```

(Figure 1)

(Figure 2)

Explanation:

The code is easiest understood when each loop is analyzed individually. Upon initial execution, the program will enter the i loop. The first step of the i loop is to enter the j loop, which then immediately triggers the k loop. The k loop will print the string “*” once for each of its four executions. Upon completion of the k loop, a line break character will be printed, constituting the end of the first iteration of the j loop. This sequence will continue another two times until the j loop is completed. This will trigger the output of another line break character - resulting in an empty line. This entire process will be repeated another four times until the entire program has finished execution. This can be easily visualized using Figure 2 (above), where the state of each variable is labelled.