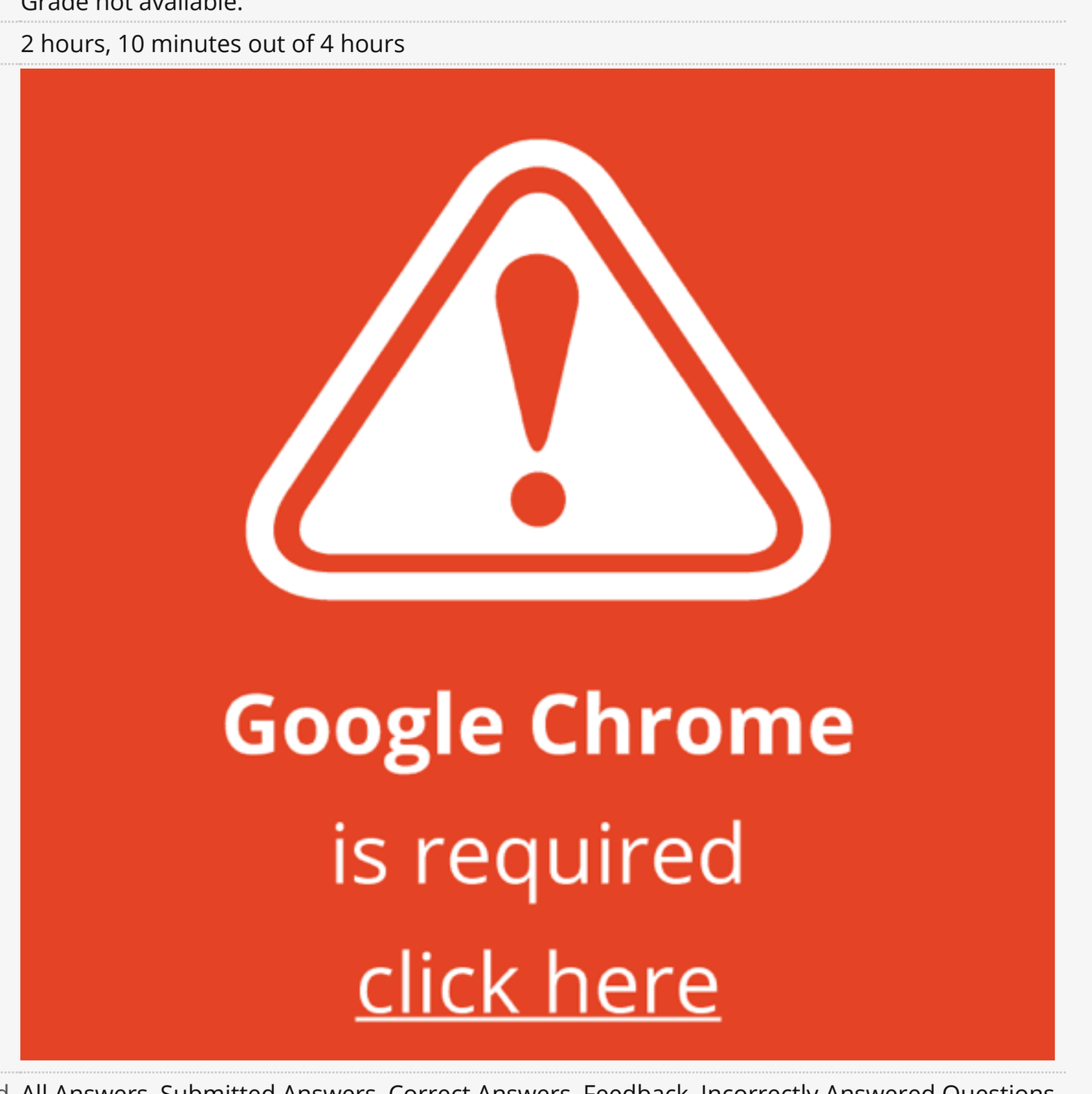
Review Test Submission: Sample Exam 1 (Secure Browser)

User	Rod Miles Simms
Course	CS-1714-0B1-Fall-2020-Computer Programming II
Test	Sample Exam 1 (Secure Browser)
Started	9/30/20 11:34 AM
Submitted	9/30/20 1:45 PM
Status	Needs Grading
Attempt Score	Grade not available.
Time Elapsed	2 hours, 10 minutes out of 4 hours
Instructions	
Results Displayed	All Answers, Submitted Answers, Correct Answers, Feedback, Incorrectly Answered Questions

Question 1

0 out of 4 points

What is the output of the following code?

int numbers[] = { 35, 57, 78, 66, 41, 12 };

int *ptrA = numbers;

int *ptrB = ptrA + 2;

int *ptrC = ptrB + 1;

```
printf( "%d, %d, %d\n", *ptrA, *ptrB, *ptrC );
```

Selected Answer: ✗ `35, 78, 57`

Answers:

`66, 78, 41`

`35, 78, 57`

`66, 35, 41`

✓ `35, 78, 66`

Question 2

0 out of 4 points

In vim, in Command mode, what does the letter j do?

Selected Answer: ✗ Moves cursor up one line

Answers:

✓ Moves cursor down one line

Moves cursor up one line

Finds the first occurrence of the letter j

Jumps to the next page

Question 3

4 out of 4 points

How are parameters passed by default when it comes to regular variables of primitive data types?

Selected Answer: ✓ Pass by value

Answers:

Pass by name

Pass by pointer

✓ Pass by value

Pass by reference

Question 4

4 out of 4 points

What is a pointer in C?

Selected Answer: ✓ A variable that holds the memory address of another variable

Answers:

A variable that holds multiple values

A special variable that can be used to represent Boolean values in C

✓ A variable that holds the memory address of another variable

A primitive data type

Question 5

0 out of 4 points

What is the output of this code?

```
#define VALUE 10;

int main( int argc, char* argv[] )
{
    int x = 3 + VALUE;
    printf( "%d", x );
    return 0;
}
```

Selected Answer: ✗ This code results in a compile-time error.

Answers:

3

This code results in a compile-time error.

✓ 13

This code results in a run-time error

Question 6

0 out of 4 points

What is the output of the following code?

```
#include <stdio.h>
int fun(int *num)
{
    return (*num)--;
}
int main(int argc, char* argv[])
{
    int num = 16;
    for(fun(&num); fun(&num); fun(&num))
        printf("%d ", fun(&num));
    return 0;
}
```

Selected Answer: ✗ Infinite loop

Answers:

13 10 7 4 1

Infinite loop

15 12 9 6 3

✓ 14 11 8 5 2

Question 7

0 out of 4 points

What is the correct statement to input a string without whitespace? Assume the string input will be stored in this variable:

```
char input[ 100 ];
```

Selected Answer: ✗ `scanf( "%s", &input );`

Answers:

`scanf( "%ls", input );`

`scanf( "%s", &input );`

`scanf( "%ls", &input );`

✓ `scanf( "%s", input );`

Question 8

4 out of 4 points

What is the format specifier for a double?

Selected Answer: ✓ %lf

Answers:

%m

✓ %lf

%d

%c

Question 9

0 out of 4 points

What are the contents of the array after this declaration and initialization?

```
int a[ 10 ] = { 5, 7, 8, 9, 11 };
```

Selected Answer: ✗ This statement causes a syntax error because the dimension does not match the number of elements given.

Answers:

`5, 7, 8, 9, 11`

✓ `5, 7, 8, 9, 11, 0, 0, 0, 0, 0`

`0, 0, 0, 0, 0, 5, 7, 8, 9, 11`

This statement causes a syntax error because the dimension does not match the number of elements given.

Question 10

4 out of 4 points

What is the output of this code?

```
int numbers[] = { 35, 57, 78, 66, 41, 12 };
int *ptrA = numbers + 3;
int *ptrB = ptrA + 1;
int *ptrC = ptrA - 1;
printf( "%d, %d, %d\n", *ptrA, *ptrB, *ptrC );
```

Selected Answer: ✓ `66, 41, 78`

Answers:

✓ `66, 41, 78`

`38, 56, 77`

`35, 57, 78`

`78, 66, 41`

Question 11

Needs Grading

Write a **function definition** that prints the following pattern, based on the value for **N** taken as an input parameter.

If N = 3, it should print

```
  *
 * *
* * *
```

If N = 4, it should print

```
   *
  * *
 * * *
* * * *
```


Selected Answer:

```
int main(int argc, char* argv[]){
  //outer loop = rows count; next loop = whitespaces; last loop = stars
  int i;
  int N;
  scanf("%d", &N);//N taken as input parameter
  int row;
  for (row = 1, row <= N; row++){
    for (i = 1; i <= N - row; i++){
      printf(" ");//print whitespace
    }
    for (i = 1; i <= 2 * row; i++){// # of stars per line
      printf("*");//
    }
    printf("\n");//newline after loop
```

Correct Answer:

```
void patternPrint(int N)
{
    int i=0, j=0, k=0;
    for(i=0; i<N; i++)
    {
        for(k=0; k<(N-i-1); k++)
            printf(" ");
        for(j=0; j<=i; j++)
            printf("* ");
        printf("\n");
    }
}
```

Response Feedback: [None Given]

Question 12

Needs Grading

Consider this block of code below. Trace the following code as explained in class. Show the starting address, the intermediate and final values of the variables. A variable of type 'int' takes 4 bytes and any pointer variable takes 8 bytes of space. Assume the starting address that is available is 1000 (calculate in decimal).

For each of the starting addresses S-1 to S-9, there is only one number, please write that down. However, for each of the variable values V-1 to V-9, there can be more than one values (starting, intermediate and final). Write all of them down, separated by comma. For example, if V-4 has 3 values then write them down as V-4 = 1, 2, 3.

You DON'T need to create a table for your answer. Just mention in each separate line, variable name = answer. For example,

S-1 = 1

S-2 = 2

and so on...

```
int X = 20, Y = 30, Z = 40;

int *p;

int arr[5] = { 6, 5, 4, 3, 2 };

p = arr;

*p++ = 10;

p = &arr[2];

*++p = 20;
```

Starting address location is 1000 (calculate in decimal)

Variable Name	Starting Address	Value
X	S-1	V-1
Y	S-2	V-2
Z	S-3	V-3
p	S-4	V-4
arr[0]	S-5	V-5
arr[1]	S-6	V-6
arr[2]	S-7	V-7
arr[3]	S-8	V-8
arr[4]	S-9	V-9

Selected Answer: [None Given]

Correct Answer:

S-1 = 1000
S-2 = 1004
S-3 = 1008
S-4 = 1012
S-5 = 1020
S-6 = 1024
S-7 = 1028
S-8 = 1032
S-9 = 1036

V-1 = 20
V-2 = 30
V-3 = 40
V-4 = 1020, 1024, 1028, 1032
V-5 = 6, 10
V-6 = 5
V-7 = 4
V-8 = 3, 20
✓ V-9 = 2

Response Feedback: [None Given]

Question 13

Needs Grading

Write code for the struct header file named "**product.h**" that consists of the following:

- header guard
- struct definition for **Product** which consists of 3 things: name (string of 10 characters), price (double value) and quantity (integer value)
- function prototypes for the following 3 functions:
  - **fillProduct()** - The function will take in two parameters: array by reference and the length of the array. The function will not return anything. The function will initialize all the **quantity** values to be equal to the **(length - index) * 10** . The **price** values will be equal to **(index * 2.5)** . All product names should be initialized to "sample".
  - **calculateAveragePrice()** - The function will take in two parameters: array by reference and the length of the array. It will calculate the average price of all products, **without** considering the quantity of each product. This average value will be returned back by the function.
  - **totalPrice()** - The function will take in three parameters: array by reference, length of the array, and the double totalPrice value passed as a pointer which will be updated with the calculated value. The function will not return anything. The **totalPrice** value should be updated to contain the weighted sum of the total price of all prices considering the quantity of each product as well.

Selected Answer: [None Given]

Correct Answer:

```
#ifndef PRODUCT_H
#define PRODUCT_H

typedef struct Product
{
    char name[10];
    double price;
    int quantity;
} Product;

void fillProducts(Product[], int);
double calculateAveragePrice(Product[], int);
void totalPrice(Product[], int, double*);

#endif;
```

Response Feedback: [None Given]

Question 14

Needs Grading

Write code for the struct source file named "**product.c**" that consists of the following:

- includes for the header files
- function definitions for the following 3 functions:
  - **fillProduct()** - The function will take in two parameters: array by reference and the length of the array. The function will not return anything. The function will initialize all the **quantity** values to be equal to the **(length - index) * 10** . The **price** values will be equal to **(index * 2.5)** . All product names should be initialized to "sample".
  - **calculateAveragePrice()** - The function will take in two parameters: array by reference and the length of the array. It will calculate the average price of all products, **without** considering the quantity of each product. This average value will be returned back by the function.
  - **totalPrice()** - The function will take in three parameters: array by reference, length of the array, and the double totalPrice value passed as a pointer which will be updated with the calculated value. The function will not return anything. The **totalPrice** value should be updated to contain the weighted sum of the total price of all prices considering the quantity of each product as well.

Selected Answer: [None Given]

Correct Answer:

```
#include<stdio.h>
#include<string.h>
include"product.h"

void fillProducts(Product p[], int n)
{
    int i=0;
    for(i=0; i<n; i++)
    {
        strcpy(p[i].name, "sample");
        p[i].price = i*2.5;
        p[i].quantity = (n-i)*10;
    }
}

double calculateAveragePrice(Product p[], int n)
{
    double sum=0;
    int i=0;
    for(i=0; i<n; i++)
    {
        sum += p[i].price;
    }
    return sum/n;
}

void totalPrice(Product p[], int n, double* totalPrice)
{
    *totalPrice = 0;
    int i=0;
    for(i=0; i<n; i++)
    {
        *totalPrice += (p[i].price * p[i].quantity);
    }
}
```

Response Feedback: [None Given]

Question 15

Needs Grading (Extra Credit)

Consider that the above Product files are already present and you are writing the code below in the main() function. Write code to for the following:

- Create an integer N, which will be the number of products.
- Take the value of N from the user.
- Dynamically allocate memory for an array of N Product objects.

Selected Answer: [None Given]

Correct Answer:

```
int n=0;
scanf("%d", &n);
Product* p = (Product*)malloc(sizeof(Product)*n);
```

Response Feedback: [None Given]

Thursday, October 1, 2020 7:19:42 PM CDT

← OK