**BTP100: Final Exam Review**

1. State two reasons why structured programming insists on one entry point and one exit point to any construct in a program such as for-next, while, a function, etc… (Week 3 – Computations: Logic)
2. What are two ways a function can change and return a variable? (Week 6 – Modularity: Pointers)
3. When passing parameters to functions, what is pass by value and pass by address? When would you use pass by value and when would you use pass by address? (Week 6 – Modularity: Pointers)
4. With reference to functions, what is meant by
   1. Modular Design (give three design principles)?
   2. Cohesion? (Week 6 – Modularity: Functions)
5. Why might you use const in a function parameter list? (Week 8 – Modularity: Functions, Arrays and Structures)
6. What is printed out in the following example if the user enters ‘3’?

What might the return value of factorial() represent?

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

int factorial(int, double \*);

main() {

int num;

double res;

printf("Enter a number : ");

scanf("%d", &num);

int retVal = factorial(num, &res);

if(retVal == 0) {

printf("The factorial of %d is %.2lf\n", num, res);

}

return 0;

}

int factorial(int number, double \*result) {

\*result = 1;

for (int i = 1; i <= number; ++i) {

\*result = \*result \* (double)i;

}

return 0;

}

[SEE Review1.c]

1. What does the following program print out:

#define \_CRT\_SECURE\_NO\_WARNINGS

#define CURRENT\_YEAR 2019

#define NUM 3

#include <stdio.h>

struct Car {

char brand[31];

int year;

double purchasePrice;

double currentValue;

};

int GetValue(struct Car \*);

int main() {

struct Car car[NUM] = {"Toyota Corolla", 2008, 21000.00, 21000.00,

"Mazda 3", 2013, 24000.00, 24000.00,

"Honda Accord", 2016, 30000.00, 30000.00 };

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

int err = GetValue(&car[i]);

if (err == 0) {

printf("The %d %s was bought for %.2lf but is now worth %.2lf\n",

car[i].year, car[i].brand,

car[i].purchasePrice, car[i].currentValue);

}

else {

printf("Your %d %s is worthless!\n", car[i].year, car[i].brand);

}

}

return 0;

}

int GetValue(struct Car \*car) {

int retVal=0;

double depreciation = 3000.00 \* (CURRENT\_YEAR - car->year);//Loses $3000.00/year

car->currentValue = car->purchasePrice - depreciation;

if (car->currentValue <= 0) {

car->currentValue = 0;

retVal = -1;

}

return retVal;

}

[SEE Review2.c]

1. Write a function called CurveMarks that takes the array of students and multiplies each mark by CURVE\_RATIO. If a mark is over 100, set the mark to 100

#define \_CRT\_SECURE\_NO\_WARNINGS

#define CURVE\_RATIO 1.2

#define NUM 5

#define NUM\_MARKS 5

#include <stdio.h>

struct Student {

char name[31];

int marks[NUM\_MARKS];

int studentNumber;

};

void CurveMarks(struct Student st[]);

int main(void) {

struct Student student[NUM] = { "Bill", 45, 57, 78, 62, 56, 10000,

"Tomasz", 66,67,78,44,59, 10001,

"Anusha", 72, 76, 74, 81, 69, 10002,

"Peng", 61, 62, 47, 56, 59, 10003,

"Erika", 77, 72, 42, 66, 79, 10004 };

CurveMarks(student);

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

printf("%s's marks are now", student[i].name);

for (int j = 0; j < NUM\_MARKS; ++j) printf(" %d", student[i].marks[j]);

printf("\n");

}

return 0;

}

[SEE: Review3.c]

1. There are five errors in the following code. Can you find them?

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

int ReturnGrade(const double mark);

int main(void) {

double markInPercent;

do {

printf("Enter your mark in percent (enter -1 to quit): ");

scanf("%lf", markInPercent);

if (markInPercent >= '0') {

char grade[] = ReturnGrade(markInPercent);

switch (grade) {

case 'A':

printf("You got an A. You win a scholarship of $5000\n");

case 'B':

printf("You got a B. You win a scholarship of $1000\n");

case 'C':

case 'D'://fall-through

printf("You got a %c. You passed\n", grade);

break;

case 'F':

printf("You have failed.\n");

}

printf("\n");

}

} while (markInPercent >= 0);

}

char ReturnGrade(const double mark) {

char grade;

mark += 2;//One lab was not fair, add 2 percent to each mark

if (mark >= 80) grade = 'A';

else if (mark >= 70) grade = 'B';

else if (mark >= 60) grade = 'C';

else if (mark >= 50) grade = 'D';

else grade = 'F';

return grade;

}

[SEE commented out code in Errors.c]

1. Write a program that stores geological data for an earthquake. Create a structure called GeoData which stores the following elements:

Name of the city [30 characters]

Latitude of the city [double float]

Longitude of the city [double float]

Wave amplitude of the tremours [double float]

Richter Scale of the earthquake[double float]

The main program will store geological data for three cities of type *GeoData*. The main program will prompt the user to enter the following for each of the three cities:

The city name, (assume only one string in the name)

The latitude and longitude separated by a space on the same line

The amplitude of the wave in millimeters

The main program will calculate the richter scale for each entry by calling a function CalculateRichterScale which will return an integer and will accept a pointer to a GeoData structure.

The main function will determine which city has the greatest tremor. The main function will then print out which city has the greatest tremor, its latitude and longitude, the amplitude of the shock waves, and the richter scale of the earthquake.

The function CalculateRichterScale return an integer (0:no error, -1: error). It takes the wave amplitude of the tremor and calculates the richter scale according to the following formula:

richterScale = sqrt(amplitude);

To use the sqrt() function, you will have to include the math library in math.h.

[SEE: Review4.c]