Cape Peninsula University of Technology

Faculty of Engineering and the Built Environment
Department of Electrical, Electronic and Computer Engineering

Software Design 1 (SDN150S)

Due Date: April 17, 2024.
CLASS TEST 1 (MEMO)

1. Complete the C program below to convert temperatures from Fahrenheit to Celsius for a range of temperatures from 0°F to 300°F with a step of 20°F using a for loop. (**10 Marks**)

```
#include <stdio.h>
int main() {
  float Fahrenheit, celsius;
  int lower = 0, upper = 300, step = 20;

  Fahrenheit = lower;
  printf("Fahrenheit Celsius\n");
  for(Fahrenheit = lower; Fahrenheit <= upper; Fahrenheit = Fahrenheit + step) {
    celsius = (5.0 / 9.0) * (Fahrenheit - 32.0);
    printf("%10.0f %7.1f\n", Fahrenheit, celsius);
  }
  return 0;
}
```

2. Fill in the **blank sections** to complete a C program that calculates the sum of the first N natural number. (**8 Marks**)

```
#include <stdio.h>
int main() {
  int n, sum = 0;
  printf("Enter a positive integer: ");
  scanf("%d", &n);

  for (int i = 1; i <= n; i++) {
    sum += i;
  }
  printf("Sum of the first %d natural numbers is: %d\n", n, sum);
return 0;
}
```

3. You are part of the cybersecurity team tasked with developing a security monitoring system. The system is designed to assess the security status of critical CPUT infrastructure and respond to different levels of threats automatically. Your current objective is to enhance the alert system to identify and categorize various security breaches based on their severity.

   Instruction:
   Complete the **blank** sections on the provided C code to implement a multi-level security breach detection system using only nested conditional statements. The system should first check for unauthorized access, if true, then assess for system tampering, if true, and finally determine the extent of network intrusion, taking appropriate actions based on the severity of the situation. (**20 Marks**)

```c
#include <stdio.h>

void check_security_status(int access_alert, int tamper_alert, int
network_intrusion) {
  printf("CPUT System Security Check Initiated:\n");
  if (access_alert == 1) {
    printf("Access Secure\n");
  } else {
    printf("Security Breach: Unauthorized Access\nPerforming System Check...\n");
    if (tamper_alert == 0) {
      printf("Security Breach: No Tampering Detected\n");
    } else {
      printf("Security Breach: Tampering Detected\n");
      if (network_intrusion != 0) {
        printf("Security Breach: Intrusion Detected\n");
        if (network_intrusion >= 1 && network_intrusion <= 3) {
          printf("Security Breach: Severity Level 1 to 2 Possibly Compromised\n");
        } else {
          printf("Security Breach: Shutting Down Mainframe!!!\nGoodbye\n");
        }
      }
    }
  }
}

int main() {
  int access_alert = 0; // 1 if no unauthorized access is detected, else it is
  int tamper_alert = 1; // 1 if tampering is detected, 0 otherwise
  int network_intrusion = 5; // Represents the level of network intrusion detected

  check_security_status(access_alert, tamper_alert, network_intrusion);
  return 0;
}
```

4. You are a software developer working at a technology company that specializes in educational tools. The company is currently developing an interactive application that helps students learn basic mathematics through various exercises. One of the features you are tasked with is creating a feature that generates multiplication tables.

   Instructions:

   You are provided with a partial codebase that is intended to generate a multiplication table based on two input variables. Complete the **blank** sections of the following code to generate a correct multiplication table for the given inputs. (**20 Marks**)

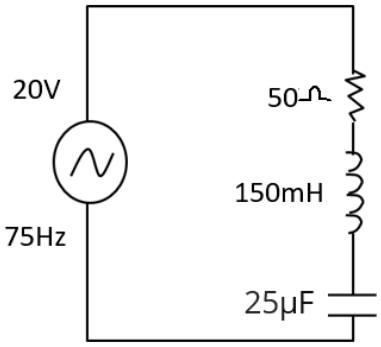
```c
#include <stdio.h>
#include <stdlib.h>

// Function definition
int* generateMultiplicationTable(int n, int size) {
  int* table = (int*)malloc(size * sizeof(int)); // Memory allocation
  if(table == NULL){
    printf("Memory allocation failed\n");
    exit(1);
  }
  for(int i=0; i<size; i++) {
    table[i] = n * (i + 1);
  }
  return table;
}

int main(){
// local variable initialzation
  int n = 3; int m = 10; // input values
int j = 0;
  int* table = generateMultiplicationTable(n, m);
  printf("Multiplication table for %d:\n", n);
```

```
  while(j < m){
    printf("%d x %d = %d\n", n, j+1, table[j]);
    j++;
  }
  free(table);
  return 0;
}
```

5. You are designing a series RLC circuit for a laboratory test involving an AC signal application. The circuit consists of a 25µF (i.e., 25 x 10-6) capacitor, a 0.15 H inductor, a 50 ohm resistor, and a 20V AC signal source operating at a frequency of 75 Hz.



Write a C program that incorporates functions to accomplish the following tasks:
a) Calculate the capacitive reactance and the inductive reactance of the circuit.
b) Determine the overall impedance of the circuit.
c) Compute the rms (root mean square) current flowing through the circuit.
d) Calculate the voltage drop across the resistor, the inductor, and the capacitor separately.
e) Calculate the total power consumption in the circuit.
f) Determine the resonant frequency of the circuit.

Instructions:
Make sure to structure your code using appropriate function definitions and calls for each required calculation. Use comments to explain the purpose of each function and segment of your code. Ensure your program is well-organized and easy to understand, as this will be part of your grading criteria. Submit your main.c file here. (**20 Marks**)

```
#include <stdio.h>
#include <math.h>
#define M_PI 3.141593   // define pi

// Function prototypes
double calculateCapacitiveReactance(double frequency, double capacitance);
double calculateInductiveReactance(double frequency, double inductance);
double calculateImpedance(double resistance, double Xc, double XL);
double calculateRmsCurrent(double voltage, double impedance);
double calculateVoltage(double current, double reactance);
double calculatePower(double current, double resistance);
double calculateResonantFrequency(double inductance, double capacitance);

int main() {
    // Declare input variables
    double frequency = 75.0; // in Hz
    double capacitance = 25e-6; // in Farads (25µF)
    double inductance = 150e-3; // in Henrys (150mH)
    double resistance = 50.0; // in Ohms
    double voltage = 20.0; // in Volts (RMS)

    // Function call for each expression
    double Xc = calculateCapacitiveReactance(frequency, capacitance);
```

```
    double XL = calculateInductiveReactance(frequency, inductance);
    double impedance = calculateImpedance(resistance, Xc, XL);
    double rmsCurrent = calculateRmsCurrent(voltage, impedance);
    double voltageAcrossResistor = calculateVoltage(rmsCurrent, resistance);
    double voltageAcrossInductor = calculateVoltage(rmsCurrent, XL);
    double voltageAcrossCapacitor = calculateVoltage(rmsCurrent, Xc);
    double powerConsumed = calculatePower(rmsCurrent, resistance);
    double resonantFrequency = calculateResonantFrequency(inductance, capacitance);

    // Output result to console
    printf("Capacitive Reactance (Xc): %lf Ohms\n", Xc);
    printf("Inductive Reactance (XL): %lf Ohms\n", XL);
    printf("Impedance (Z): %lf Ohms\n", impedance);
    printf("RMS Current (I): %lf A\n", rmsCurrent);
    printf("Voltage across Resistor: %lf V\n", voltageAcrossResistor);
    printf("Voltage across Inductor: %lf V\n", voltageAcrossInductor);
    printf("Voltage across Capacitor: %lf V\n", voltageAcrossCapacitor);
    printf("Power Consumed: %lf W\n", powerConsumed);
    printf("Resonant Frequency: %lf Hz\n", resonantFrequency);

    return 0;
}

// Function definitions
double calculateCapacitiveReactance(double frequency, double capacitance) {
    return 1.0 / (2 * M_PI * frequency * capacitance);
}

double calculateInductiveReactance(double frequency, double inductance) {
    return 2 * M_PI * frequency * inductance;
}

double calculateImpedance(double resistance, double Xc, double XL) {
    return sqrt(pow(resistance, 2) + pow(XL - Xc, 2));
}

double calculateRmsCurrent(double voltage, double impedance) {
    return voltage / impedance;
}

double calculateVoltage(double current, double reactance) {
    return current * reactance;
}

double calculatePower(double current, double resistance) {
    return pow(current, 2) * resistance;
}

double calculateResonantFrequency(double inductance, double capacitance) {
    return 1.0 / (2 * M_PI * sqrt(inductance * capacitance));
}
```

6. In C, which operator is used to combine two conditions? (**2 Marks**)
   a) &&
   b) ==
   c) ||
   d) ++

7. How do you modify the for loop below into an infinite loop? Select all that apply.
   `for(int i = 0; i < 10; i++)` (**2 Marks**)
   `{ /* statements */ }`

   a) Change i < 10 to i <= 10
   b) Change i < 10 to i >= 0
   c) Correct: Remove i < 10 completely

d) Correct: Change i++ to i--

8. Which of the following is not a valid way to declare a function in C? (**2 Marks**)
   a) function_name(int x, int y)
   b) void function_name(int x, int y)
   c) char function_name(int x, int y)
   d) int function_name(int x, int y)

9. Match the following terms with their definitions: (**8 Marks**)

Software	A collection of programs, data, and instructions that tell a computer how to perform specific tasks.
Hardware	The physical components of a computer system that can be seen and touched.
Algorithm	A step-by-step procedure or formula for solving a problem or achieving a desired result.
Compiler	A program that translates source code written in a high-level language into machine code that can be executed directly by a computer.

10. Match the following terms with their definitions: (**8 Marks**)

Syntax	The set of rules that define the combinations of symbols that are considered to be correctly structured programs or expressions.
Loop	A control flow statement that allows a code block to be executed repeatedly based on a given condition.
Conditional Statement	A statement that performs different actions based on whether a certain condition is true or false.
Function	A named block of code that performs a specific task and can be used multiple times in a program.