MMT-001: Programming and Data Structures

Session 1: Familiarisation with compiling process

The aim of this session and the next session is to familiarise the student with the process of compiling and running C programs.

Program 1

Modify the program in unit1-prog1.c so that it outputs

```
This is very easy!!
```

Program 2

Modifying the values in the file unit1-prog8.c to the following, compiling, running and seeing the output:

1. x = 12, y = 13 and x = 11, y = 14
2. x = 1.2, y = 4.1 and x = 0.2 and y = 5

Program 3

1. Calculating the integral $\int_0^1 \frac{1}{1+x^2} dx$ by actual integration and comparing it with values obtained from the program unit1-simpsonf.c.

3. Changing the function to a) x^2 b) e^x and comparing the values given by the program with the values got by actual integration.

Session 2: Familiarisation with compiling process

Program 1

Inserting `printf()` statements in unit3-prog1.c page 53, Block 1, compiling and running it to check the answer.

Program 2

Compiling and running the file unit3-prog2.c in page 54 of Block 2 to check the answer for the exercise E3) of Unit 3.

Program 3

Completing the program unit3-prog0.c in page 71 of Block 1, which is given as the solution to exercise E1) of Unit 3, by giving values for the remaining 3 cases and compiling and running the program to check the output.

Program 4

Modifying the program in unit3-prog0.c so that it reads the value of a and b using scanf() instead of including the values in the program itself, modify it so that

it reads the values of a and b using `scanf`(). Checking the program by compiling and running it with 4 different values of a and b covering all the 4 cases.

Program 5

Writing a program that prompts the user for basic pay and calculates and prints basic +

HRA(30% of basic)+DA(46% of basic).

Session 3

Program 1

Writing a program similar to unit2-ex7ans.c in page 45 of Block 1that checks for overflow in multiplication when the two operands are **unsigned long** and when they are **long double**. The program should also print the overflow when it occurs.

Program 2

Writing a program that prompts for a natural number n and prints the nth Fibonacci

number. It should also print an error message if the Fibonacci number is bigger than

`ULONG_MAX`.

Program 3

Writing a program that find all primes less than `ULONG_MAX` and saves it in a file called primes.

Session 4

Writing a C function to determine the number of days between two dates passed to it.

Session 5

Program 1

Writing a function that evaluates the polynomial by Horner's method. Horner's method is as follows: If

$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$,the Horner's method is given by the equation

$$f(x) = ((\ldots((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0)$$

It should prompt for the coefficients of the polynomial, store the values in an array, prompt the value for which the polynomial is to be evaluated and evaluate and print the value. The function should be able to evaluate any polynomial of degree less than 20. The function should be checked by writing a program that calls it.

Program 2

Writing functions similar to Horner's method for evaluating $f_o(x)$and $f_e(x)$ where $f_o(x)$ and $f_e(x)$ are defined as follows: If $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, then

$$f_o = \sum_{\substack{0 \le i \le n \\ i\ odd}} a_i x^i, f_e(x) = \sum_{\substack{0 \le i \le n \\ i\ even}} a_i x^i$$

Note that $f(a) = f_e(a) + f_o(a)$and $f(-a) = f_e(a) - f_o(a)$. It is easy to show that, if the polynomial $f(x)$has integer coefficients and has a rational root $\frac{p}{q}$, then p $|a_0$and q$|a_n$. In particular, it has an integer root a, then a $|a_o$. Writing a function for checking whether a polynomial $f(x)$ has integer roots by finding the value of $f(\pm a)$ for all divisors of a of a_o

Program 3

Writing a program that computes the value of $\tan^{-1} x$ using the power series expansion

$$x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots$$

Session 6

Program to evaluate definite integrals of a function for arbitrary number of intervals using Simpson's one third rule.

Session 7

Program 1

Program for solving equations by bisection method.

Program 2

Program for solving algebraic equations by false position(regula falsi) method.

Session 8

Program to solve an intermediate value problem by Runge-Kutta method

Session 9

Program 1

Currency converter program using **switch ( )**

Program 2

Program for finding maximum number of real roots using Descartes Rule of Signs.

Session 10

Program 1

Program for binary powering algorithm

Program 2

Program for finding the order of $a \in \frac{\boldsymbol{Z}}{p\boldsymbol{Z}}$ for a prime p.

Session 11

Program 1

Program for linear regression

Program 2

Program to fit a transcendental curve of the form $Y = aX^b$ for a given set of data.

Session 12

Program for creating a n × n array of float using calloc().

Session 13

Program for finding the inverse of a matrix using Gauss-Jordan method.

Session 14

Program for working with rational number using **struct.**

Session 15

Program that maintains a telephone directory in a file.

Session 16

Program for administering an objective type test.

Session 17

Program to create a linked list which holds the names of the cities, sorts the list and prints the list.

Session 18

Program 1

Program for reversing a string using a stack.

Program 2

Program to check whether the brackets are properly matched.

Session 19

Program for implementing a queue containing integers as a linked list by implementing functions for creation of the queue, enqueue and dequeue.

Session 20

Program to create a binary search tree, insert words, do an in-order, pre-order and post-order traversals of the trees and list the keys in each node traversed.