Operating Systems Reflective Journal

Note: You **must** provide evidence for your answers, where appropriate, by inserting examples of Practical work you undertook; for eg: solutions attempted, code written.
You **must** also provide, at the end of this document, your answers to the questions from the Practical Handout.
Please answer **all** questions.

Name Danyil Tymchuk	**Student ID** B00167321
Practical Name/Topic GNU Emacs and C Programming	**Session Dates** 27/02/25
What did I read for this session (apart from the Practical handout)?	Tutorialspoint
What were the main points that were new to me in the Practical?	Emacs, I already knew the basic of programmind in C before this
What were two main things I learned?	Should be Ecams and C Programming
In trying to solve the problems, what were the things I tried?	Write pseudocode
What went wrong in my initial attempts to solve each problem, and why?	Tried to put the main() {} beffore another function that I have to call in the main block
What went right in my final attempts to solve each problem, and why?	Place main() {} below, because the pc should read that function before call it
How did my learning in the Practical relate to what I'm learning in other parts of the module; for example: in the Lecture	Ok
What did we not cover that I expected we should?	Idk

What was new or surprising to me?	Nothing yet
I am still unsure about…	No
Issues that interested me a lot, and that I would like to study in more detail	C Programming, Commands
What I most liked about this Practical was…	C Programming
What I most disliked about this Practical was…	Emacs
How did you feel about how you performed in the Practical?	Good
Miscellaneous interesting facts I learned in this Practical…	None

Please insert your answers from the Practical Handout here (use as many pages as you need):

Key Sequances:

- Find a file: (C-x C-f)
- Save a file: (C-x C-s)
- Move to the end of a line: (C-e)
- Move to the beginning of a line: (C-a)
- Undo: (C-x u)
- Split the screen into two windiws: (C-x 2)
- Quit Emacs: (C-x C-c)

Command in 4.1:

- `$ gcc -v`
- `gcc version 12.3.0 (Mageia 12.3.0-3.mga9)`

`Describe three options:`

- `-o <output_file> – This option allows you to specify the output file name instead of the default.`
  - `gcc program.c -o my_program – This compiles program.c and creates an executable named my_program.`

- -c – Compiles source files into object files without linking.
  - gcc -c program.c – This produces program.o, which can be linked later with other object files.
- -Wall – Enables most common warnings to help detect potential issues in the code.
  - gcc -Wall program.c -o program – This helps catch syntax errors, unused variables, and other potential problems.

hello.c

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     printf("Hello, World!\n");
6.     return 0;
7. }
8.
```

- $ gcc hello.c -o sayhi
- $ ./sayhi

Pseudo-code for Looping Program:

1. Start
2. Initialize a loop to run 20 times
3. Print "This is iteration X" (where X is the loop iteration number)
4. End loop
5. Print "Goodbye"
6. Stop

- loopprinter1.c

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     for (int i = 1; i <= 20; i++) {
6.         printf("This is iteration %d\n", i);
7.     }
8.     printf("Goodbye\n");
9.     return 0;
10.  }
11.
```

Pseudo-code for Looping with Function:

1. Start
2. Define a function to execute the loop and print iterations
3. Call function from main()
4. Print "Goodbye" from main()
5. Stop

- loopprinter2.c

```
1. #include <stdio.h>
2.
```

```
3. void printIterations(int n)
4. {
5.     for (int i = 1; i <= n; i++) {
6.         printf("This is iteration %d\n", i);
7.     }
8. }
9.
10.  int main()
11.  {
12.      printIterations(20);
13.      printf("Goodbye\n");
14.      return 0;
15.  }
16.
```

Pseudo-code for Array Program:

1. Start
2. Declare an integer array of size 10
3. Loop from 0 to 9, assigning array[i] = i*i
4. Print each element
5. Stop

- mysimplearray.c

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int array[10];
6.     for (int i = 0; i < 10; i++) {
7.         array[i] = i*i;
8.         printf("%d\n", array[i]);
9.     }
10.      return 0;
11.  }
12.
```

Pseudo-code for Array Sum Program:

1. Start
2. Declare an integer array of size 10
3. Initialize a sum variable to 0
4. Loop from 0 to 9, assigning array[i] = i*i and adding to sum
5. Print each element
6. Print sum
7. Stop

- calcSum.c

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int array[10], sum = 0;
6.     for (int i = 0; i < 10; i++) {
```

```
7.          array[i] = i*i;
8.          sum += array[i];
9.          printf("%d\n", array[i]);
10.       }
11.       printf("Sum: %d\n", sum);
12.       return 0;
13.   }
14.
```