



## UNIVERSITY COLLEGE TATI (UCTATI)

## FINAL EXAMINATION QUESTION BOOKLET

COURSE CODE	: FEP 1023
COURSE	: INTRODUCTION TO PROGRAMMING
SEMESTER/SESSION	: 3 - 2024/2025
DURATION	: 3 HOURS

Instructions:


1. This booklet contains **4** questions. Answer **ALL** questions.
2. All answers should be written in answer booklet.
3. Write legibly and draw sketches wherever required.
4. If in doubt, raise up your hands and ask the invigilator.

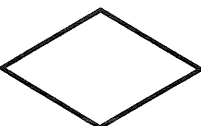
**DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO**

**THIS BOOKLET CONTAINS 11 PRINTED PAGES INCLUDING COVER PAGE**


QUESTION 1

a) Identify the name of the following flowchart block:

i.  ..... (1 mark)

ii.  ..... (1 mark)

iii.  ..... (1 mark)

iv.  ..... (1 mark)

b) Based on circuit diagram given at Figure 1, write declaration for each connected pin using #define method.

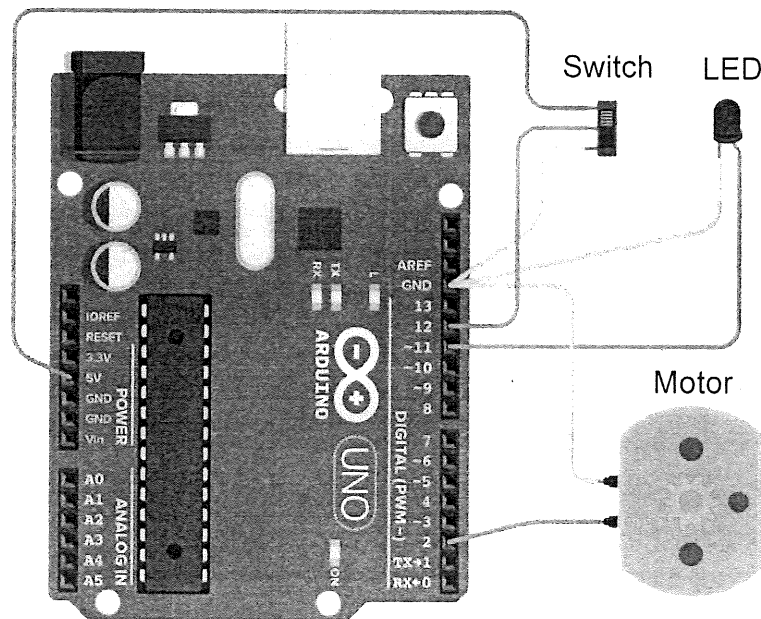


Figure 1: Circuit diagram of input and output component

(3 marks)

INTRODUCTION TO PROGRAMMING (FEP 1023)

---

c) Use suitable 'Data Types' to declare the following variable:

- i. `x = -1` (1 mark)
- ii. `speed = 128` (1 mark)
- iii. `weight = 15.3` (1 mark)

d) State the name for each of the operators below:

- i. `<=` (1 mark)
- ii. `//` (1 mark)

e) The program code given below has a syntax error. One (1) error has been identified (*circled*). Discover & circle another five (5) syntax error.

```
#define led = 9
unsigned char motor 3;

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(Motor, OUTPUT);
}

void loop()
{
  digitalWrite(led, 1);
  delay(1000;
  digitalWrite(motor, 1);
  delay(1000);
}
```

(5 marks)

## QUESTION 2

- a) Based on flow chart given in Figure 2, produce a program code using `'while'` loop method.

(16 marks)

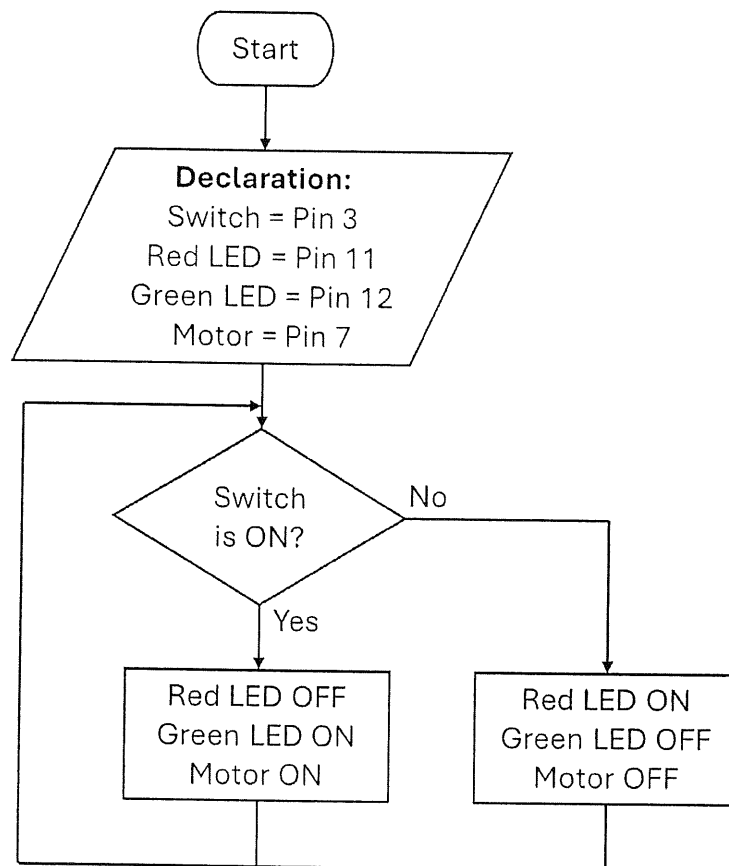


Figure 2

- b) Produce a `'for'` loop program format to count from 0 until 99.

(5 marks)

## INTRODUCTION TO PROGRAMMING (FEP 1023)

c) Based on flow chart given in Figure 3, produce a program code using '*if...if*' method. Pin connected are as follows:

- S1 at pin 10
- S2 at pin 11
- LED (D1) at pin 4
- LED (D2) at pin 5

(12 marks)

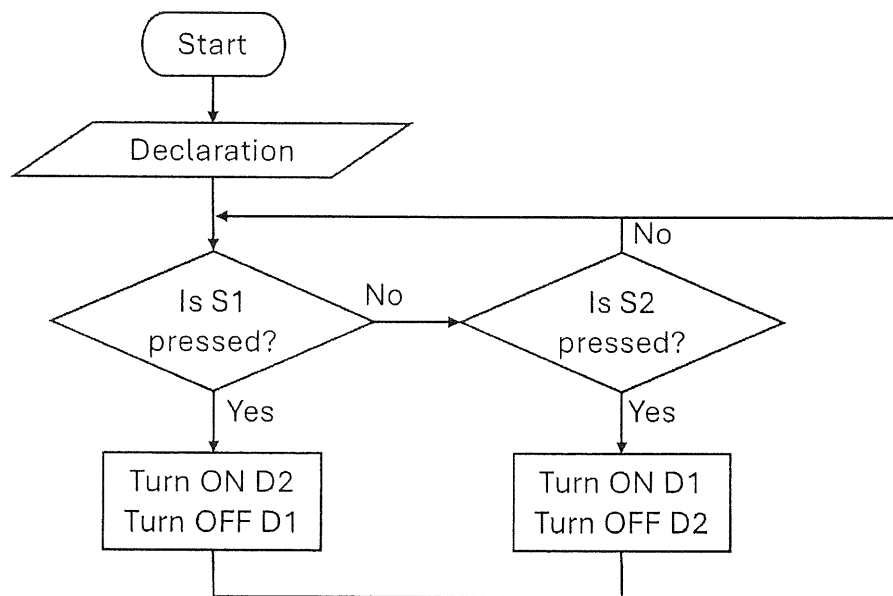


Figure 3

## INTRODUCTION TO PROGRAMMING (FEP 1023)

## QUESTION 3

- a) Based on array declaration given below, show the data stored (index value) for the *array* address shown at Table 1.

```
int pin[8]={5, 6, 7, 8, 9, 10, 11, 12};
int corak[]={0b10000000,
             0b11000000,
             0b11100000,
             0b11110000,
             0b11111000,
             0b11111100,
             0b11111110,
             0b11111111};
int corak2[4]={0b11110000,0b00001111,0b11000011,0b00111100};
```

Table 1

Array address	Index value
pin[2]	i.
pin[8]	ii.
corak[1]	iii.
corak[7]	iv.
corak2[1]	v.
corak2[3]	vi.

(1 mark)

(1 mark)

(1 mark)

(1 mark)

(1 mark)

(1 mark)

- b) Based on array declaration given below, show your step to solve the following mathematical operation below (*Neglect the value change after the mathematical operation for each question*).

```
int nombor[]={5, 1, 103, 0, 0, 27, 9, 3, 7};
```

i.  $nombor[3] = nombor[1] + nombor[5]$

(2 marks)

ii.  $nombor[4] = nombor[5] / nombor[7]$

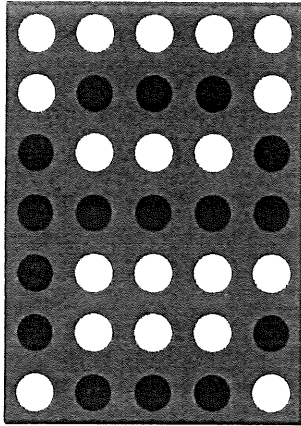
(2 marks)

iii.  $nombor[0] = nombor[4] - nombor[8]$

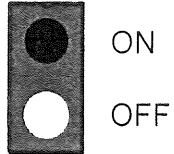
(2 marks)

c) Produce an *array* declaration for the LED Matrix display given below.

i.

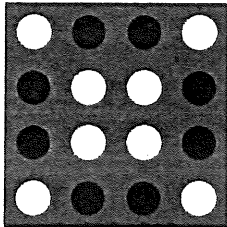


LED State



(8 marks)

ii.



LED State



(5 marks)

**QUESTION 4**

a) Write a suitable *function declaration* (Prototype) for the following statement:

- i. LED on. (1 mark)
- ii. LED blinking. (1 mark)
- iii. Motor ON Clockwise. (1 mark)
- iv. Set mode for input and output. (1 mark)
- v. All Output OFF (1 mark)

b) Based on '*function*' program code below, show the return value for '*y*' at the end of the program when '*c*' value given as follows:

```
int kecerunan(int c, int x)
{
    int m = 3;
    int y = m*x + c;    //y=mx+c
    return y;
}
```

- i. When, kecerunan(2, 3); (3 marks)
- ii. When, kecerunan(11, 4); (3 marks)

INTRODUCTION TO PROGRAMMING (FEP 1023)

- c) Based on circuit given in Figure 4, produce a program to turn ON LED with two different pattern (given in Table 2) of LED light movement using '*Function*' method.

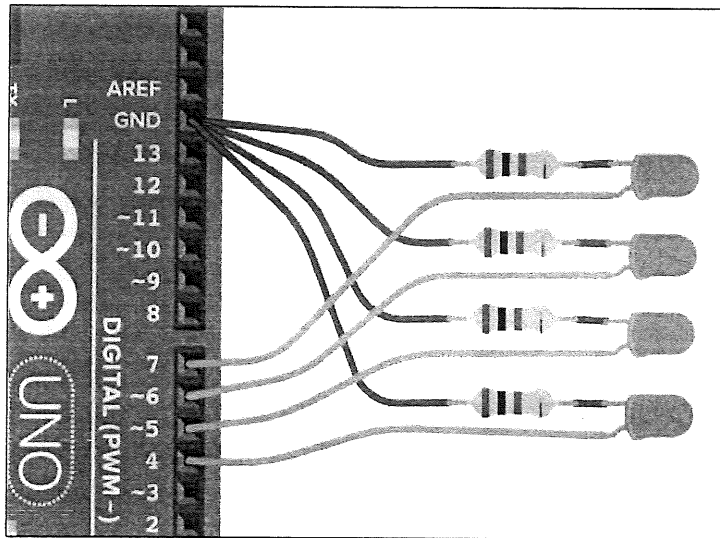


Figure 4

Table 2

Pattern 1	Pattern 2	
↓	↑	LED Movement Direction

(14 marks)

-----End of question-----

## INTRODUCTION TO PROGRAMMING (FEP 1023)

**ATTACHMENT 1 : CODE FORMATING****Declaration**

```
#define led1 12
int suis = 3;
int x,y;
```

**Array Declaration**

```
int kakiLED[]={4,5,6,7,8,9,10};
int ledMatrix[5][4]={
    {0,1,1,0},
    {1,0,0,1},
    {1,0,0,1},
    {1,0,0,1},
    {0,1,1,0},
};
```

**Prototype**

```
void pattern_1();
void padam(int i);
```

**Setup Function**

```
void setup()
{
    pinMode(led1,OUTPUT);
    pinMode(suis,INPUT);
}
```

**Main Program (Main Function)**

```
void loop()
{
    pattern_1();
    padam(1000); } Call function
}
```

**Function / Sub-Program / Sub-Function**

```
void pattern_1()
{
}
```

**Selection**

```
if(digitalRead(suis)==1)
{
}
else
{
}
```

```
-----
if(digitalRead(suis1)==1)
{
}
if(digitalRead(suis2)==1)
{
}
-----
if(digitalRead(suis1)==1)
{
}
else if(digitalRead(suis2)==1)
{
}
else
{
}
```

**Looping**

```
while(digitalRead(suis)==1)
{
}
```

```
-----
do
{
}
```

```
while(digitalRead(suis)==1)
```

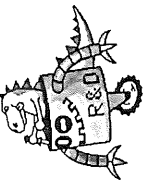
```
-----
for(int i=0; i<8; i++)
{
}
```

**Array**

```
for(int i=0; i<4; i++)
{
    digitalWrite(kakiLED[i],1);
}
```

INTRODUCTION TO PROGRAMMING (FEP 1023)

ATTACHMENT 2 : CHEAT SHEET



## ARDUINO CHEAT SHEET V.02c

Mostly taken from the extended reference:  
<http://arduino.cc/en/Reference/Extended>  
 Gavin Smith - Robots and Dinosaurs, The Sydney Hackspace

Flash (2x, for bootloaders)	16kB	128kB	128kB	128kB	128kB
SRAM	1kB	2kB	8kB	8kB	8kB
EEPROM	512B	1kB	1kB	1kB	1kB

Microcontroller	Pinouts	Max I/O
Atmega168	54 + 16 analog	54 + 16 analog
Atmega328P	54 + 16 analog	54 + 16 analog
Atmega1280	54 + 16 analog	54 + 16 analog

Pin	Function
0 - GND	Ground
1 - TX	Transmit
2 - RX	Receive
3 - INT0	Interrupt
4 - GND	Ground
5 - VCC	Power
6 - AREF	Analog Reference
7 - GND	Ground
8 - A0	Analog Input
9 - TX	Transmit
10 - RX	Receive
11 - INT1	Interrupt
12 - GND	Ground
13 - VCC	Power
14 - A1	Analog Input
15 - TX	Transmit
16 - RX	Receive
17 - INT2	Interrupt
18 - GND	Ground
19 - VCC	Power
20 - A2	Analog Input
21 - TX	Transmit
22 - RX	Receive
23 - INT3	Interrupt
24 - GND	Ground
25 - VCC	Power
26 - A3	Analog Input
27 - TX	Transmit
28 - RX	Receive
29 - INT4	Interrupt
30 - GND	Ground
31 - VCC	Power
32 - A4	Analog Input
33 - TX	Transmit
34 - RX	Receive
35 - INT5	Interrupt
36 - GND	Ground
37 - VCC	Power
38 - A5	Analog Input
39 - TX	Transmit
40 - RX	Receive
41 - INT6	Interrupt
42 - GND	Ground
43 - VCC	Power
44 - A6	Analog Input
45 - TX	Transmit
46 - RX	Receive
47 - INT7	Interrupt
48 - GND	Ground
49 - VCC	Power
50 - A7	Analog Input
51 - TX	Transmit
52 - RX	Receive
53 - INT8	Interrupt
54 - GND	Ground
55 - VCC	Power
56 - A8	Analog Input
57 - TX	Transmit
58 - RX	Receive
59 - INT9	Interrupt
60 - GND	Ground
61 - VCC	Power
62 - A9	Analog Input
63 - TX	Transmit
64 - RX	Receive
65 - INT10	Interrupt
66 - GND	Ground
67 - VCC	Power
68 - A10	Analog Input
69 - TX	Transmit
70 - RX	Receive
71 - INT11	Interrupt
72 - GND	Ground
73 - VCC	Power
74 - A11	Analog Input
75 - TX	Transmit
76 - RX	Receive
77 - INT12	Interrupt
78 - GND	Ground
79 - VCC	Power
80 - A12	Analog Input
81 - TX	Transmit
82 - RX	Receive
83 - INT13	Interrupt
84 - GND	Ground
85 - VCC	Power
86 - A13	Analog Input
87 - TX	Transmit
88 - RX	Receive
89 - INT14	Interrupt
90 - GND	Ground
91 - VCC	Power
92 - A14	Analog Input
93 - TX	Transmit
94 - RX	Receive
95 - INT15	Interrupt
96 - GND	Ground
97 - VCC	Power
98 - A15	Analog Input
99 - TX	Transmit
100 - RX	Receive

**Structure**  
void setup() void loop()

**Control Structures**  
if (x<5) { } else { }  
switch (myvar) {  
case 1:  
break;  
case 2:  
break;  
default:  
}  
for (int i=0; i<=255; i++) { }  
while (x<5) { }  
continue; //Go to next in do/for/while loop  
return x; //Or 'return;' for voids  
goto //considered harmful :-)

**Further Syntax**  
// (single line comment)  
/\* (multi-line comment) \*/  
#define DOZEN 12 //Not baker's!  
#include <avr/pgmspace.h>

**General Operators**  
= (assignment operator)  
+ (addition) - (subtraction)  
\* (multiplication) / (division)  
% (modulo)  
== (equal to) != (not equal to)  
< (less than) > (greater than)  
<= (less than or equal to)  
>= (greater than or equal to)  
&& (and) || (or) ! (not)

**Pointer Access**  
& (reference operator)  
\* (dereference operator)

**Bitwise Operators**  
& (bitwise and) | (bitwise or)  
^ (bitwise xor) ~ (bitwise not)  
<< (bitshift left) >> (bitshift right)

**Compound Operators**  
++ (increment) -- (decrement)  
+= (compound addition)  
-= (compound subtraction)  
\*= (compound multiplication)  
/= (compound division)  
&= (compound bitwise and)  
|= (compound bitwise or)

**Qualifiers**  
static // persists between calls  
volatile // use RAM (nice for ISR)  
const // make read-only  
PROGRAM\_MEMORY // use flash

**Digital I/O**  
pinMode(pin, [INPUT|OUTPUT])  
digitalWrite(pin, value)  
int digitalWrite(pin)  
//Write High to inputs to use pull-up res

**Analog I/O**  
analogReference(DEFAULT|INTERNAL|EXTERNAL)  
int analogRead(pin) //Call twice if switching pins from high Z source.  
analogWrite(pin, value) // PWM

**Advanced I/O**  
tone(pin, freqHz)  
noTone(pin)  
shiftOut(dataPin, clockPin, [MSBFIRST|LSBFIRST], value)  
// unsigned long pulse(pin, [HIGH|LOW])

**Time**  
unsigned long millis() // 50 days overflow.  
unsigned long micros() // 70 min overflow  
delay(ms)  
delayMicroseconds(us)

**Math**  
min(x, y) max(x, y) abs(x)  
constrain(x, minval, maxval)  
map(val, fromL, fromH, toL, toH)  
pow(base, exponent) sqrt(x)  
sin(rad) cos(rad) tan(rad)

**Random Numbers**  
randomSeed(seed) // Long or int  
long random(max)  
long random(min, max)

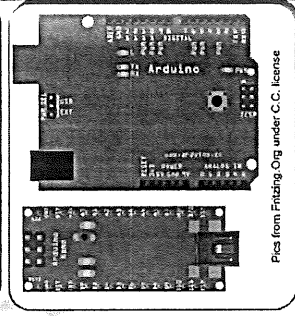
**Bits and Bytes**  
lowByte() highByte()  
bitRead(x, bin) bitWrite(x, bin, bit)  
bitSet(x, bin) bitClear(x, bin)  
bit(bint) //bin: 0-LSB 7-MSB

**Data Types**  
void  
boolean (0, 1, false, true)  
char (e.g. 'a', -128 to 127)  
unsigned char (0 to 255)  
byte (0 to 255)  
int (-32,768 to 32,767)  
unsigned int (0 to 65535)  
word (0 to 65535)  
long (-2,147,483,648 to 2,147,483,647)  
unsigned long (0 to 4,294,967,295)  
float (-3.4028235E+38 to 3.4028235E+38)  
double (currently same as float)  
sizeof(myint) // returns 2 bytes

**Strings**  
char S[15];  
char S2[8] = "a","b","c","d","e","f";  
char S3[8] = {"a","b","c","d","e","f","g","h"};  
//included \0 null termination  
char S4[] = "arduino";  
char S5[8] = "arduino";  
char S6[15] = "arduino";

**Arrays**  
int myInts[6];  
int myPlus[] = {2, 4, 8, 3, 6};  
int mySensVals[6] = {2, 4, -8, -3, 2};

**Conversion**  
byte() word() float()



Pics from Pinning.Orig under C.C. license

