University of Mila
Faculty of Science and Technology
Course: Structure of Computers and Applications

Academic Year: 2025/2026
1st year ST - ENG & LMD
Semester 1

Date: January 13th, 2026. Time: 15:00 – 17:00 Duration: 02.00 h

Model Answer for the Regular session exam

EXERCISE 01 (03 points)

1. Write an algorithm to find and display all even numbers from 0 to a given positive integer N. Use the conventional form. ··· (02 pts)
2. Verify your result by creating a trace table that shows each step of your algorithm's execution for N = 5. ··· (01 pt)

Solution 1: Algorithm to find and display all even numbers from 0 to N ··· (02 pts)

```
Step1: Start
Step2: Input N
Step3: for I = 0 to N . Step 1
Step4: is (I %2=0) ?
       If true, Then continue
       if false, Then go to Step6
Step5: Print I
Step6: Next I
Step7: end
```

Points breakdown for algorithm:

- Step 1 (Start): 0.125 pt
- Step 2 (Input N): 0.25 pt
- Step 3 (For loop initialization): 0.25 pt
- Step 4 (Condition check): 0.5 pt
- Step 5 (Print statement): 0.5 pt
- Step 6 (Loop increment): 0.25 pt
- Step 7 (End): 0.125 pt
- Total: 2.0 pts

Solution 2: Trace Table for N = 5 ··· (01 pts)

Step	N	i	i <= N	i % 2 == 0	Display	i = i + 1
1	5	0	True	True	0	1
2	5	1	True	False	-	2
3	5	2	True	True	2	3
4	5	3	True	False	-	4
5	5	4	True	True	4	5
6	5	5	True	False	-	6
7	5	6	False	-	-	-

Points breakdown for trace table:

- Correct table structure: 0.125 pt
- Each correct iteration (rows 1-6): 0.1 pt each (total 0.5 pt)
- Correct termination condition (row 7): 0.125 pt
- Correct output values: 0.25 pt
- Total: 1.0 pt

Output: 0, 2, 4

EXERCISE 02 (05 points)

1. Write an algorithm and flowchart for the absolute value function. ··· (04 pts)
2. Briefly explain how your algorithm handles both positive and negative inputs. ··· (01 pt)

1.1) Algorithm (1st method): Calculate Absolute Value Function ··· **(02 pt)**

Step 1: Start (0.125 pt)

Step 2: Read the input value (x) (0.25 pt)

Step 3: Is x $\geq$ 0? (0.5 pt)

- If true, then continue
- If false, then go to step 5

Step 4: Calculate y $\leftarrow$ x, go to step 6 (0.5 pt)

Step 5: Calculate y $\leftarrow -$x (0.25 pt)

Step 6: Print "y(", x, ") = ", y (0.25 pt)

Step 7: Stop (0.125 pt)

Algorithm (2nd method): Calculate Absolute Value Function ··· **(02 pt)**

Step 1: Start (0.125 pt)

Step 2: Read the input value (x) (0.25 pt)

Step 3: If $x \geq 0$ then (0.5 pt)

Step 3.1: Calculate $y \leftarrow x$ (0.25 pt)

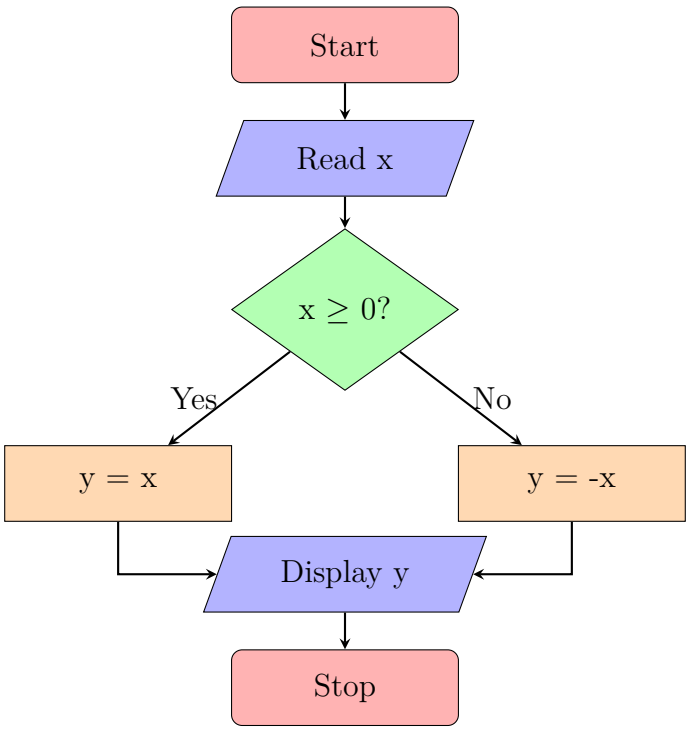
Step 4: Else (0.25 pt)

Step 4.1: Calculate $y \leftarrow -x$ (0.25 pt)

Step 5: Display "y(", x, ") = ", y (0.25 pt)

Step 6: Stop (0.125 pt)

1.2) Flowchart for Absolute Value Function ··· **(02 pts)**



Points breakdown for flowchart:

- Correct symbols (start/stop, input/output): 0.5 pt
- Decision diamond with condition: 0.4 pt
- Process boxes for both cases: 0.4 pt
- Correct arrows and flow: 0.4 pt
- Yes/No labels on decision paths: 0.3 pt
- Total: 2.0 pts

2. Explanation of how the algorithm handles positive and negative inputs: ··· (01 pt)

- **For positive inputs (x ≥ 0):**
  - The algorithm takes the "Yes" path from the decision node
  - Sets y = x (no change)
  - Example: If x = 5, then y = 5
- **For negative inputs (x < 0):**
  - The algorithm takes the "No" path from the decision node
  - Sets y = -x (negates the negative to get positive)
  - Example: If x = -3, then y = -(-3) = 3

EXERCISE 03 (06 points)

1. Write a C program that adds two integers and returns their sum. ··· (02 pts)
2. Write another C program that: ··· (03 pts)
   a) Accepts two float numbers and one integer from the user
   b) Calculates their sum and average
   c) Prints the results
3. Provide sample output for your second program. ··· (01 pt)

1. C program that adds two integers and returns their sum: ··· (02 pts)

```c
#include <stdio.h>
int main() {
    int num1 = 10;
    int num2 = 20;
    int sum = num1 + num2;
    printf("Sum: %d\n", sum);
    return 0;
}
```

Points breakdown for Program 1:

- #include directive: 0.125 pt
- main() function: 0.25 pt
- Variable declarations: 0.5 pt
- Sum calculation: 0.5 pt
- printf statement: 0.5 pt
- return 0: 0.125 pt
- Total: 2.0 pts

Output:

```
Sum: 30
```

2. C program that accepts two float numbers and one integer: ··· **(03 pts)**

```c
#include <stdio.h>
int main() {

    float num1, num2;
    int num3;
    printf("Enter two float numbers and one integer: ");
    scanf("%f %f %d", &num1, &num2, &num3);

    float total = num1 + num2 + num3;
    float average = total / 3.0;

    printf("Sum = %.2f\n", total);
    printf("Average = %.2f\n", average);

    return 0;
}
```

Points breakdown for Program 2:

- #include directive: 0.125 pt
- main() function: 0.25 pt
- Variable declarations (mixed types): 0.5 pt
- printf for input prompt: 0.0 pt
- scanf with correct format specifiers: 0.5 pt
- Total sum calculation: 0.5 pt
- Average calculation: 0.5 pt
- printf for sum with formatting: 0.25 pt
- printf for average with formatting: 0.25 pt
- return 0: 0.125 pt
- Total: 3.0 pts

3. Sample output for your second program: ··· **(01 pt)**

```
Enter two float numbers and one integer: 3.5 2.5 4
Sum = 10.00
Average = 3.33
```

EXERCISE 04 (06 points)

1. Write a complete C program that determines whether a given integer is odd or even. ··· (03 pts)
   Requirements:
   a) Read an integer from the user
   b) Use conditional statements
   c) Display "ODD" or "EVEN"
2. Modify your program to also indicate if the number ("ODD" or "EVEN") is positive or negative. ··· (03 pts)

1. Complete C program that determines whether a given integer is odd or even: ··· **(03 pts)**

```c
#include <stdio.h>

int main() {
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number % 2 == 0) {
        printf("EVEN\n");
    } else {
        printf("ODD\n");
    }

    return 0;
}
```

Points breakdown for Program 1:

- #include directive: 0.125 pt
- main() function: 0.25 pt
- Variable declaration: 0.5 pt
- printf for input prompt: 0.0 pt
- scanf for input: 0.5 pt
- if condition with modulus operator: 0.5 pt
- printf for EVEN case: 0.5 pt
- printf for ODD case: 0.5 pt
- return 0: 0.125 pt
- Total: 3.0 pts

2. Modified program to also indicate if the number ("ODD" or "EVEN") is positive or negative: ··· **(03 pts)**

```c
#include <stdio.h>

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    // Check ODD/EVEN first
    if(num % 2 == 0) {
        // Number is EVEN - check if positive or negative
        if(num > 0) {
            printf("EVEN POSITIVE\n");
        } else {
            printf("EVEN NEGATIVE\n");
        }
    }
    else {
        // Number is ODD - check if positive or negative
        if(num > 0) {
            printf("ODD POSITIVE\n");
        } else {
            printf("ODD NEGATIVE\n");
        }
    }

    return 0;
}
```

Points breakdown for Program 2:

- #include directive: 0.125 pt
- main() function: 0.125 pt
- Variable declaration: 0.125 pt
- printf/scanf for input: 0.5 pt
- Outer if for ODD/EVEN check: 0.5 pt
- Inner if for positive/negative (EVEN case): 0.5 pt
- Inner if for positive/negative (ODD case): 0.5 pt
- Correct output messages for all 4 cases: 0.5 pt
- return 0: 0.125 pt
- Total: 3.0 pts