University of Mila
Faculty of Science and Technology
Course: Introduction to Programming
Date: May 20th, 2026. Time: 10:30 – 12:30

Academic Year: 2025/2026
1st year ST - ENG & LMD
Semester 2
Duration: 02.00 h

Model Answer - Regular Session Exam

SOLUTION EXERCISE 01 (03 points):

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

[0.125 point]

```c
float findMAX(float a, float b);
```

[0.5 point (function prototype)]

```c
int main() {
    float num1, num2, maximum;
```

[0.25 point (variable declarations + main)]

```c
    printf("Enter two floating-point numbers: ");
    scanf("%f %f", &num1, &num2);
```

[0.25 point (scanf)]

```c
    maximum = findMAX(num1, num2);
```

[0.5 point (function call)]

```c
    printf("The maximum number is: %.2f\n", maximum);
    return 0; }
```

[0.25 point (printf) 0.125 point (return 0)]

```c
float findMAX(float a, float b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

[1 point (function definition)]

SOLUTION EXERCISE 02 (06 points):

Q1. (2 points) - Program Explanation:

The program does the following:

1. Asks the user to enter the size of an array (S) [0.25 pt]
2. Declares a variable-length array of size S [0.25 pt]
3. Reads S integer values from the user into the array [0.25 pt]
4. Calls `computeSum()` to calculate the sum of all elements [0.25 pt]

5. Calculates the average by dividing the sum by S [0.25 pt]

6. Displays the sum and average [0.25 pt]

Role of `computeSum()`: [0.5 point]

The `computeSum()` function takes an integer array and its size as parameters, iterates through all elements, adds each element to a running total, and returns the final sum. It does not modify the original array.

Q2. (1.5 points) - Output:

Output

```
Enter array size: 4                    //[0.25 point]
arr[0] = 12                            //[0.75 point]
arr[1] = 8
arr[0] = 12
arr[2] = 20
arr[3] = 4
Sum = 44                               //[0.25 point]
Average = 11.00                        //[0.25 point]
```

Q3. (2.5 points) - Function Prototype:

Purpose: [1.5 point]

The function prototype tells the compiler about the function's return type, name, and parameters before the function is actually defined. This allows the compiler to check if the function is called correctly in main().

What happens if removed:

- The compiler will assume the function returns an `int` (default) [0.25 pt]
- A warning message will be generated [0.25 pt]
- If the function returned a different type, it would cause errors [0.25 pt]
- The program will still compile and run correctly in this case [0.25 pt]

SOLUTION EXERCISE 03 (06 points):

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

```c
int main() {
    int n;
```

[0.25 point]

```c
    printf("Enter the size of the array: ");
    scanf("%d", &n);
```

[0.25 point]

```c
    int arr[n], posArr[n], negArr[n];
    int posCount = 0, negCount = 0;
```

[1 point (array declarations)]

```c
    // Read array elements
    printf("Enter %d elements:\n", n);
    for(int i = 0; i < n; i++) {
        printf("arr[%d] = ", i);
        scanf("%d", &arr[i]);
    }
```

[1 point (reading array elements)]

```c
    // Split into positive and negative arrays
    for(int i = 0; i < n; i++) {
        if(arr[i] > 0) {
            posArr[posCount] = arr[i];
            posCount++;
        } else {
            negArr[negCount] = arr[i];
            negCount++;
        }
    }
```

[2 points (splitting logic)]

```c
    // Display positive array
    printf("\nPositive array (%d elements):\n", posCount);
    for(int i = 0; i < posCount; i++) {
        printf("%d ", posArr[i]);
    }

    // Display negative array
    printf("\nNegative array (%d elements):\n", negCount);
    for(int i = 0; i < negCount; i++) {
        printf("%d ", negArr[i]);
    }
    return 0; }
```

[1.5 points (displaying both arrays)]

SOLUTION EXERCISE 04 (05 points):

Q1. (4 points) - Completed Program with Blanks:

```c
#include <stdio.h>
int main() {
    int A[3][2] = {{10, 20}, {30,40}, {50,60}};
    int B[2][2] = {{1, 0}, {0,1}};
    int C[3][2], i, j, k;

    // Matrix multiplication: C = A * B
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 2; j++) {
            C[i][j] = 0;
            for (k = 0; k < 2; k++) {
                C[i][j] = C[i][j] + (A[i][k] * B[k][j]);
            }
        }
```

```
    }

    // Display the result
    printf("Result matrix C:\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 2; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Blanks filled (0.5 point each = 4 points):

1. `j = 0; j < 2; j++` [0.5 pt]
2. `0` [0.5 pt]
3. `k = 0; k < 2; k++` [0.5 pt]
4. `C[i][j] + (A[i][k] * B[k][j])` [0.5 pt]
5. `i = 0; i < 3; i++` [0.5 pt]
6. `j = 0; j < 2; j++` [0.5 pt]
7. `"%d ", C[i][j]` [0.5 pt]
8. `"\n"` [0.5 pt]

Q2. (1 point) - Output:

Output of the Program

```
Result matrix C:
10 20
30 40
50 60
```