Full name: [illegible]
Matricula:
Group:

Task 1/3 (7p): Create an algorithm in UML for the following task:

- Create and fill an array that has N elements. Both the elements (a_i) and N will be read from the keyboard.
- Enforce the following rules: N > 10. Each element (a_i) must be in between -10 and 10 (boundary values not included)
- User will enter a value x, which is between -10 and 10.
- Create two new arrays: one containing the numbers below x, the second above x. Output those two new arrays.

7

Task 2/3 (6p): Transform the following code to an equivalent UML activity diagram. In addition to writing out the code statements into UML elements, also include the semantics (What is being done? What is being checked? For how long? Etc).

```c
#include <stdio.h>

#define LEN 5

int main(void)
{
    int values[LEN], validNum;
    int i, j;
    i = 0;
    while (i < LEN)
    {
        printf("Enter num %d of %d: ", i + 1, LEN);
        scanf("%d", &values[i]);
        validNum = 1;
        for (j = 0; j < i; j++)
        {
            if (values[i] == values[j])
            {
                validNum = 0;
                break;
            }
        }
        if (validNum == 1)
        {
            i++;
        }
    }
    printf("The unique numbers are: ");
    for (i = 0; i < LEN; i++)
    {
        printf("%d ", values[i]);
    }
    return 0;
}
```

5

Turn page for task 3

2

Full name:

Matricula:

Group:

Task 1/3 (8p): Create an algorithm in UML for the following task:

- N student codes and their calculated average grades are being read from the keyboard
- Enforce the following rules: N > 10 and 0 ≤ grade ≤ 5
- The list is sorted by the average grade
- Assign and output the scholarships: top 3 students get 300€ scholarships, the next 7 get 100€ and the rest don't get any.

Task 2/3 (5p): Transform the following code to an equivalent UML activity diagram. In addition to writing out the code statements into UML elements, also include the semantics (What is being done? What is being checked? For how long? Etc).

```c
#include <stdio.h>

#define LEN 9
#define NORMAL_XY 100
#define MEASUREMENTS 2

int main(void)
{
    int sizes[LEN][MEASUREMENTS] = {{101, 100}, {91, 100}, {50, 100},
                                    {102, 115}, {99, 100}, {101, 102},
                                    {99, 95}, {100, 100}, {102, 101}};
    int i;
    float tolerance, validMin, validMax;

    printf("Enter the tolerance in percentage for the cutouts\n");
    scanf("%f", &tolerance);
    validMin = NORMAL_XY * (1 - (tolerance / 100));
    validMax = NORMAL_XY * (1 + (tolerance / 100));
    printf("Tolerance min = %.2f, max = %.2f\n", validMin, validMax);
    for (i = 0; i < LEN; i++)
    {
        if (sizes[i][0] < validMin || sizes[i][0] > validMax)
        {
            printf("The detail #%d length is out of tolerance\n", i);
        }
        else if (sizes[i][1] < validMin || sizes[i][1] > validMax)
        {
            printf("The detail #%d width is out of tolerance\n", i);
        }
        else
        {
            printf("The detail #%d is within acceptable limits\n", i);
        }
    }
    return 0;
}
```

Turn page for task 3

IAX0583 Programming I
Test I

Full name: [illegible]
Matricula:
Group: MVEB

Task 1/3 (8p): Create an algorithm in UML for the following task:

- N timestamps are being read from the keyboard
- Check, that the entered timestamp matches following format HH:MM:SS (24 hour clock)
- Check, that the entered timestamp is real
- Print the one of the following conclusions next to each timestamp: “Valid format, correct time”, “Valid format, incorrect time” or “Invalid format”
- E.g. “00:00:00”, “20:50:90”, “00:00*00”, “05:a1:2b”, “ab?cd:ef” etc.

2

Task 2/3 (5p): Transform the following code to an equivalent UML activity diagram. In addition to writing out the code statements into UML elements, also include the semantics (What is being done? What is being checked? For how long? Etc).

```c
#include <stdio.h>

int main(void)
{
    int value, inputValidated;
    printf("Enter a number between 0 and 10 (inclusive)\n");
    do
    {
        inputValidated = 1;
        scanf("%d", &value);
        if (value < 0 || value > 10)
        {
            inputValidated = 0;
            printf("Input out of range (0, 10)! Try again!\n");
        }
    }
    while (!inputValidated);

    printf("The validated input is %d\n", value);

    return 0;
}
```

3

Turn page for task 3

1