1.

Rivi 17: malloc ei alusta muistiaan nollaksi, joko calloc(), memset() tai for-rakenne
Rivi 17: len arvoa ei aseteta tietueeseen
Rivi 26: Myös bf-data pitää vapauttaa
Rivi 31: Rakenteiden koot pitäisi tarkastaa ensiksi
Rivi 40: Pitäisi käyttää fprintf funktiota
Rivi 40: Pitää tulostaa '0' tai '1' eikä 0 tai 1. Korjaus + '0' tai muu vastaava rakenne.

Line 17: malloc does not initialise the memory to zero, fix with either calloc(), memset() o
Line 17: The len value is never set to the struct
Line 26: bf->data has to be freed as well
Line 31: The lengths of the structures should be checked first
Line 40: The function fprintf should be used instead of printf
Line 40: The function should print '0' or '1' and not 0 or 1. Fix with + '0' or some other s

2.

```
#include <stdio.h>

void readInput(FILE *input)
{
    int value, min, max, sum = 0;
    size_t count = 0;

    while (fscanf(input, "%d", &value))
    {
        if (count++ == 0)
            min = max = value;
        if (min > value)
            min = value;
        if (max < value)
            max = value;
        sum+=value;

    }
    printf("%d %d %.2f", min, max, sum / (double)count);
}
```

3.

B, C, C, B, A, A, B, C, B, A, B, C

4.

```
#include <stddef.h>
#include <stdlib.h>

typedef struct
{
    int angles[3];
} Triangle;

int checkAngles(const int *angles)
{
    int sum = 0;
    for (size_t i = 0;i < 3;i++)
        sum+=angles[i];
    return sum == 180;
}

size_t filterTriangles(Triangle **target, const Triangle *data, size_t len)
{
    size_t count = 0;

    for (size_t i = 0;i < len;i++)
```

```
        if (checkAngles(data[i].angles))
            count++;

    Triangle *copy = malloc(sizeof(Triangle) * count);
    count = 0;

    for (size_t i = 0;i < len;i++)
        if (checkAngles(data[i].angles))
            copy[count++] = data[i];

    *target = copy;
    return count;
}
```

5.

Ensimmäisessä funktiossa on tahaton virhe, jonka takia funktio tekee määrittelemättömän toiminnon, eli kaikki vastaukset funktion toiminnasta ovat yhtä oikein.

Toinen funktio laskee syötemerkkijonossa päällä olevien bittien määrän.

Kolmas funktio muuttaa kaikki merkkijonon kirjaimet isoiksi kirjaimiksi.

The first function contains an involuntary mistake which causes the function to exhibit undefined behaviour. Therefore any description of the function is as valid or as correct as any other description.

The second function counts the number of bits that are on in the given string.

The third function changes all characters to uppercase in the string.

6.

```
double (*getOperation(char operator))(double, double)
{
    switch (operator)
    {
    case '+':
        return plus;
    case '-':
        return minus;
    case '*':
        return mul;
    case '/':
        return divd;
    default:
        return NULL;
    }
}
```

Tai / Or

```
typedef double (*type)(double, double);
type getOperation(char operator) { sama / same }
```