Programming 2 (SS 2023)
Saarland University
Faculty MI
Compiler Design Lab

Exercise Sheet

Prof. Dr. Sebastian Hack
Pascal Lauer, M. Sc.
Marcel Ullrich, B. Sc.

The exam preparation sheet has been created by the Prog2 tutors. The exercises were selected, as the tutors classified them as as potential exam questions or as good preparation. **This does not guarantee that this kind of question will be found in the exam!** The extent of this sheet considerably exeeds the exam's extent. As most questions are completely new, it is possible that mistakes crept in. Please ask in the forum, if you are uncertain a solution is correct. We will try to fix it as fast as possible.

Exercise 1: Multiple Choice

For each of the following statements decide if they are true or false. If you think that a statement is false try to explain to yourself why it is false.

1. Mips: The assembly directive `.data` marks the beginning of the code segment.
2. The Mips calling convention specifies that the stack pointer and the s-registers need to be restored by the called function before jumping back.
3. To load a word its address needs to be divisible by 4.
4. `int foo(int x);` is a valid function declaration in C.
5. In C it holds `sizeof(char)==2`.
6. `a == b` if `a.equals(b)`
7. Dividing by zero is undefined behaviour in Java.
8. It is not possible to create an object of an abstract class in Java.
9. The load factor of a hash table can be computed by dividing the number of inserted elements by the size of the hash table.
10. To run through every position in a hash table using quadratic probing it is not important to use a surjective hash function.
11. Black Box tests try to achieve full branch, path and code coverage.
12. `true` is the weakest statement.
13. $[a0] = \langle a \rangle \cdot 2$

Exercise 2:

Write a function `scream` that gets the address to a (null-terminated) ASCII string in `$a0` and prints this string in upper-case. You may assume that the string only contains letters.
Screaming is not just about saying it in upper-case, you also have to adjust your facial expressions and so on. For that reason, you should use our own `print_char` function which handles all of this. You can just give it a character in `$a0` and it will print it for you. Notice that it does more than that, so it uses lots of registers.

Example: Calling the function with `Hello World` results in printing `HELLO WORLD`.

Hint: The ASCII code of "A" is 65 and the ASCII code of "a" is 97.

Here is a start for your code. Notice, that you should only write the *TODO* part.

```
.text
main:
    ...
    jal     scream
    ...

scream:
    # TODO
    jr      $ra

print_char:
    ...
```

Exercise 3: Edit distance

The Levenshtein distance is a measure for the difference between two sequences. It is defined as the minimum number of single-character edits (insertions, deletions, substitutions) required to transform one sequence into another:

$$lev(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ lev(tail(a), tail(b)) & \text{if } a[0] = b[0], \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b) \end{cases} & \text{otherwise} \end{cases}$$

where *tail* is a substring excluding the first character, and $x[n]$ is the nth character beginning at 0.
The implementation below uses dynamic programming to cache previously computed edit distances between the prefixes of the first sequence, and the prefixes of the second sequence.
Sequences are represented as byte strings, and the cache matrix is stored in row-major format as an array of words.

1. Fill in the gaps such that `min` returns the minimum of its three arguments.

```
# Calculates the minimum of three numbers.
# @param $a0 The first number.
# @param $a1 The second number.
# @param $a2 The third number.
# @return $v0 The smallest of the three numbers.
min:
    move    $v0 $a0
    ble     ___ ___  min__second_is_greater
    move    ___ ___
    min__second_is_greater:
    ble     ___ ___ min__third_is_greater
    move    ___ ___
    min__third_is_greater:
    jr      $ra
```

2. Next, complete the following function that initializes the cache matrix. Since any sequence can be transformed into an empty one by deleting all characters, the first row and column are set to the number of characters of the substring at that position.

```
# Initializes a given edit distance matrix.
# The elements of the first row/column are
# set to their respective column/row index.
# @param $a0 The address of the matrix (stored in row-major order).
# @param $a1 The number of rows of the matrix.
# @param $a2 The number of columns of the matrix.
initialize_matrix:
    # initialize the first row to [0, $a2)
    ____    $t0 0
    ____    initialize_matrix__first_row_loop_condition
    initialize_matrix__first_row_loop_body:
        sw      $t0 ($a0)
        addiu   $a0 $a0 _
        addiu   $t0 $t0 _
    initialize_matrix__first_row_loop_condition:
        blt     $t0 ___ initialize_matrix__first_row_loop_body

    # initialize the first column to [0, $a1)
    li      $t0 _
    mulu    $t1 $a2 _
    b       initialize_matrix__first_column_loop_condition
    initialize_matrix__first_column_loop_body:
        sw      $t0 ($a0)
        _____   $a0 $a0 $t1
        _____   $t0 $t0 1
    initialize_matrix__first_column_loop_condition:
        blt     $t0 ___ initialize_matrix__first_column_loop_body

    jr      $ra
```

3. Finally, fill out the missing parts such that `distance` adheres to the calling convention.

```
# Calculates the edit distance between two strings.
# @param $a0 The address of the first string.
# @param $a1 The length of the first string.
# @param $a2 The address of the second string.
# @param $a3 The length of the second string.
# @param 0($sp) The address of the result matrix.
# @return $v0 The edit distance between the first and the second string.
distance:
    # build stack frame
    addiu   ___ ___ ___
    sw      ___ 0($sp)
    sw      ___ 4($sp)
    sw      ___ 8($sp)
    sw      ___ 12($sp)
    sw      ___ 16($sp)
    sw      ___ 20($sp)
    sw      ___ 24($sp)
    sw      ___ 28($sp)
    sw      ___ 32($sp)

    # save arguments
    move    $s0 $a0
    addiu   $s1 $a1 1
    move    $s2 $a2
    addiu   $s3 $a3 1
    lw      $s4 36($sp)

    # initialize matrix
    move    $a0 $s4
    move    $a1 $s1
    move    $a2 $s3
    jal     initialize_matrix

    # fill matrix
    li      $s5 1
    mulu    $t0 $s3 4
    addu    $s7 $s4 $t0
    b       distance__outer_fill_loop_condition
    distance__outer_fill_loop_body:
        li      $s6 1
        b       distance__inner_fill_loop_condition
        distance__inner_fill_loop_body:
            addu    $t0 $s0 $s5
            lbu     $t0 -1($t0)
            addu    $t1 $s2 $s6
            lbu     $t1 -1($t1)
            sne     $t0 $t0 $t1

            mulu    $t1 $s3 4
            subu    $t1 $s7 $t1

            lw      $a0 ($s7)
            addiu   $a0 $a0 1

            lw      $a1 4($t1)
            addiu   $a1 $a1 1

            lw      $a2 ($t1)
            addu    $a2 $a2 $t0

            jal     min
            sw      $v0 4($s7)

            addiu   $s6 $s6 1
            addiu   $s7 $s7 4
        distance__inner_fill_loop_condition:
            blt     $s6 $s3 distance__inner_fill_loop_body
            addiu   $s5 $s5 1
            addiu   $s7 $s7 4
    distance__outer_fill_loop_condition:
        blt     $s5 $s1 distance__outer_fill_loop_body

    lw      $v0 -4($s7)

    # reset stack frame
    lw      ___ 0($sp)
    lw      ___ 4($sp)
    lw      ___ 8($sp)
    lw      ___ 12($sp)
    lw      ___ 16($sp)
    lw      ___ 20($sp)
    lw      ___ 24($sp)
    lw      ___ 28($sp)
    lw      ___ 32($sp)
    addiu   ___ ___ ___

    jr      $ra
```

Exercise 4: RLE

Implement a function called `rle` that performs a run-length encoding of a given array, which replaces consecutive occurrences of the same element in an array with the count of occurrences and the value of the element.

`rle` should take a pointer `arr` to the first element of an array of `int`s and the length of this array `len` as input. You can assume that the array only contains non-negative numbers. If a number x, occurs consecutively $n \geq 2$ times in the array, `rle` should replace the n entries with two entries: $-n$ and x.

Example:

input:	1	1	1	8	4	4	4	4
output:	-3	1	8	-4	4	?	?	?

`rle` should modify the input array in-place and return the length of the modified array. As a result of the modification, the remaining entries should shift forward. In that example above, the new length is 5. It is not necessary to free the memory of the freed entries, and it doesn't matter which numbers are present in the elements beyond the new end of the array (indicated by ? in the example).

Hint: *It's unnecessary (but allowed) to use the dynamic memory allocation. If you used it, you have to deallocate the dynamically allocated memory.*

Exercise 5: Pointerchase

Determine by hand what output the following program will provide. Use the computer only to verify that your result is correct.

```c
#include <stdio.h>

int main() {
    int i = 4;
    int *ap, *bp;
    int **app, **bpp;

    int r[4] = {1,2,3,4};

    ap = &r[0];
    bp = &r[3];

    app = &ap;
    bpp = &bp;

    *ap = i;
    *(*(bpp)) = 11;

    while (*ap < 6) {
        bp--;
        if(*ap % 2 == 0) {
            *(*bpp) = *ap + 25;
        } else {
            *bp = 7;
        }
        (*ap)++;
    }

    printf("%d-%d-%d-%d\n", r[0], r[1], r[2], r[3]);
}
```

Exercise 6:

1. Draw the AST for the following `C0` program.

```
y = &x;
{
  if (x > 5)
    x = 10;
  else
    x = 0;

  while(x > 10)
    {*y = *y - 1;}
}
```

2. Consider the following program in `C0pb`:

```
{
    int x;
    int* ptx;

    x = 3;
    ptx = &x;

    while (x) {
        int** y;
        x = x - 1;
        y = &ptx;
        *ptx = -2 + **y;
    }
}
```

   (a) Type check the program using the type checking rules known from the lecture.

   (b) Execute the program by using the execution rules for C0pb on the state $\sigma = (\{\}; \{\})$.

Exercise 7:

Take a look at the following program:

```
int check(int a, int b, int c) {
    int x = 2;

    if (a > 0) {
        x += 42;
    }

    if (b % 4 == 0) {
        x += 4;
    }

    if (c != 0) {
        int y = c * x;

        if (y >= 0) {
            return y;
        } else {
            return x;
        }
    } else {
        return 0;
    }
}
```

1. Write a test suite that has a *statement coverage* of 100%, i.e. all statements are covered by the test suite.
2. Extend your test suite, such that it has a *branch coverage* of 100%. If you think that no additional tests are required for this, explain in a sentence, why this is the case.
3. Extend your test suite again, such that it has a *path coverage* of 100%. If you think that no additional tests are required for this, explain in a sentence, why this is the case.

Now, look at the following program:

```
int check2(int x, int y) {
    int v = x;
    if (v < 0) {
        v = -v;
    }
    if (v == 1337) {
        return 1;
    } else if (v > 1337) {
        if (y < 5) {
            return 2;
        }
        return 3;
    } else {
        return 0;
    }
}
```

4. Find out, what the *statement* coverage of each of the following calls to the function is. Write your answer as a fraction of the amount of covered statements and the amount of statements in the program.
   - `check2(-1337, 42)`
   - `check2(111, 6)`
   - `check2(2000, 6)`
   - `check2(1337, 2)`
   - `check2(-4321, 4)`
   - `check2(-1, 4000)`

Exercise 8:

You have been hired by the university to implement part of the Mensa dishes in Java. Some dishes have already been implemented, so part of the infrastructure already exists (like the `Ingredients` but also the abstract class `MensaDish`).
Your task is to implement the following infrastructure, avoiding any kind of code duplication.
Create an interface called `Block`, which is implemented by the abstract class `MensaBlockDish`, which also inherits from the abstract class `MensaDish`.
Make sure that the following points are met:

1. A `Block` should always be able to return its volume (the volume in cubic centimetres should be represented here with an integer)

2. A `MensaBlockDish` has beside it's volume the same attriubutes as `MensaDish`

3. `NoodleCasserole`, `PotatoCasserole`, `PotatoGratin` and `Schales` are `MensaBlockDish`'es

4. The constructors of `NoodleCasserole`, `Schales`, `PotatoCasserole` and `PotatoGratin` should not take any paramenters

5. The respective volumes and prices are:
   `NoodleCasserole`: 480 cm^3, $2,85$€
   `PotatoCasserole`: 440 cm^3, $3,15$€
   `PotatoGratin`: 400 cm^3, $3,35$€
   `Schales`: 450 cm^3, $3,45$€

6. The dishes contain the following ingredients:
   `NoodleCasserole`: pasta, cream, cheese, eggs and onions
   `PotatoCasserole`: potatoes, onions, milk and eggs
   `PotatoGratin`: potatoes, cream, milk, butter, cheese and garlic
   `Schales`: potatoes, bacon and eggs

7. The `NoodleCasserole` tastes "bad", the `PotatoCasserole` tastes "good", the `PotatoGratin` is "very tasty" and the `Schales` are "edible"

8. You should be able to take a bite out of a `MensaBlockDish`, where 18 cm^3 are eaten.
   The method `bite` should print "finished" if the meal is empty after the bite and it should print "cry" if an attempt is made to take a bite from an empty meal.

9. Also, you were asked to add the functionality that you can print the ingredients from any `MensaDish`.
   The method should be named `printIngredients` and return the ingredients = {ingr_1, ..., ingr_n} as the string "ingr_1, ..., ingr_n"

```java
public enum Ingredient {
    Pasta("pasta"),
    Potatoes("potatoes"),
    Milk("milk"),
    Cheese("cheese"),
    Cream("cream"),
    Onions("onions"),
    Butter("butter"),
    Eggs("eggs"),
    Bacon("bacon"),
    Garlic("garlic");

    private String name;

    Ingredient(String name) {
        this.name = name;
    }

    public String toString() {
        return name;
    }
}
```

```java
public abstract class MensaDish {
    protected List<Ingredient> ingredients = new LinkedList<>();
    protected double price;
    protected String name;

    protected MensaDish(String name, double price,
                        List<Ingredient> ingredients) {
```

```
        this.name = name;
        this.price = price;
        this.ingredients.addAll(ingredients);
    }

    // should print how tasty the dish is
    public abstract void taste();

    public double getPrice() {
        return this.price;
    }

    public void eat() {
        System.out.println("eating " + name);
    }







}
```

Exercise 9: Inheritance Hierarchy

Which method is called in each case? State the output of the program.
Note: If you are unsure how to solve this exercise, you can find a detailed explanation in the forum.

Class Hierarchy

```java
public class A {
  public void fun(B b) {
    System.out.println("A.fun(B)");
  }
  public void fun(D d) {
    System.out.println("A.fun(D)");
  }
}
```

```java
public class B extends A {
  public void fun(A a) {
    System.out.println("B.fun(A)");
  }
  public void fun(C c) {
    System.out.println("B.fun(C)");
  }
  public void fun(E e) {
    System.out.println("B.fun(E)");
  }
}
```

```java
public class C extends B {
  public void fun(B b) {
    System.out.println("C.fun(B)");
  }
}
```

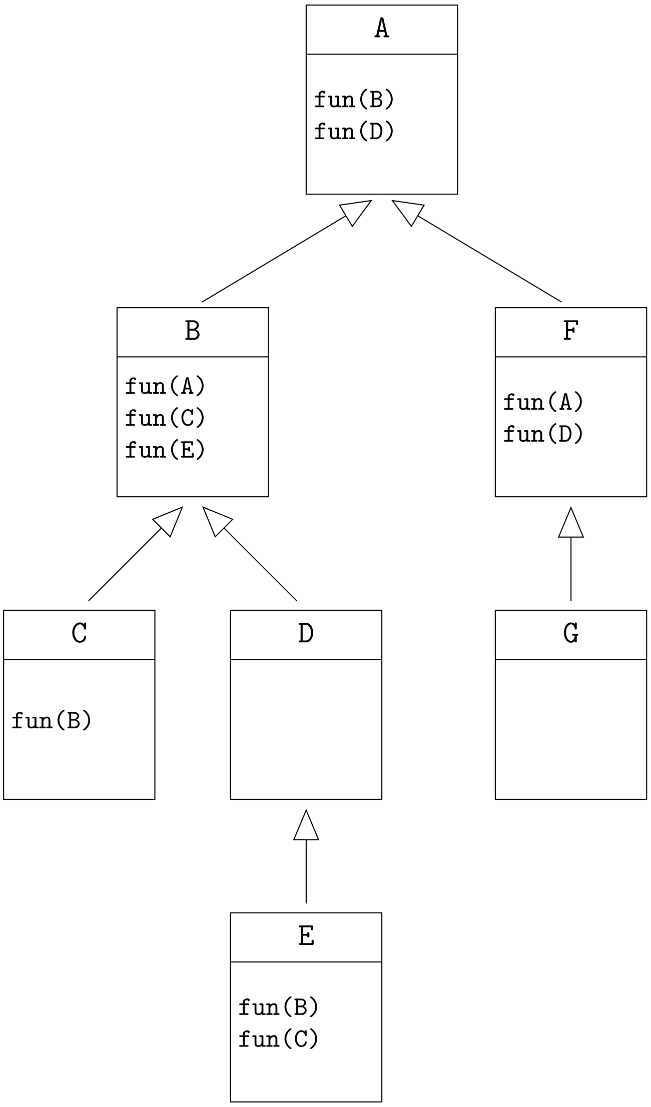
```java
public class D extends B {
}
```

```java
public class E extends D {
  public void fun(B b) {
    System.out.println("E.fun(B)");
  }
  public void fun(C c) {
    System.out.println("E.fun(C)");
  }
}
```

```java
public class F extends A {
  public void fun(A a) {
    System.out.println("F.fun(A)");
  }
  public void fun(D d) {
    System.out.println("F.fun(D)");
  }
}
```

```java
public class G extends F {
}
```

```java
public class Main {
    public static void main(
        String[] args) {
        A aa = new A();
        A ab = new B();
        A ac = new C();
        A ag = new G();
        A af = new F();
        B bb = new B();
        B bc = new C();
        B bd = new D();
        B be = new E();
        C cc = new C();
        D dd = new D();
        D de = new E();
        E ee = new E();
        F ff = new F();
        F fg = new G();
        G gg = new G();

        aa.fun(bd);
        ac.fun(bb);
        ac.fun(be);
        af.fun(dd);
        af.fun(ab);
        ag.fun(cc);

        bb.fun(ab);
        bb.fun(bc);
        bc.fun(bb);
        bc.fun(ff);
        bc.fun(dd);
        bd.fun(cc);
        be.fun(bc);
        be.fun(cc);
        be.fun(dd);

        cc.fun(de);
        cc.fun(ee);

        dd.fun(dd);
        dd.fun(bd);
        dd.fun(fg);

        ee.fun(ee);
        ee.fun(cc);

        ff.fun(fg);
        gg.fun(de);
        gg.fun(bc);
    }
}
```



Exercise 10:

Konrad Klug likes to read. The problem is that over the years he has accumulated so many books that he now has a hard time finding them in his bookshelves. To solve this problem, he has created the following class and implemented a hash function. Here, `digitSum(int x)` calculates the digit sum of `x`.

```java
public class Book {

    private String title;
    private String author;
    private int ISBN;

    public int hashCode() {
        return digitSum(ISBN);
    }
}
```

The following is a table of some of Konrad's books that he wants to hash:

Title	**Author**	**ISBN**
The Lord of the Rings	J. R. R. Tolkien	978-0544273443
The Art of Computer Programming	Donald E. Knuth	978-0137935109
The Hobbit	J. R. R. Tolkien	978-0261103344
The Ultimate Hitchhiker's Guide to the Galaxy	Douglas Adams	978-0345453747
Elements of Causal Inference	J. Peters, D. Janzing, B. Schölkopf	978-0262037310
The Chronicles of Narnia	C. S. Lewis	978-0066238500
Sherlock Holmes	Arthur Conan Doyle	978-0141040288
Artificial Intelligence: A Modern Approach	Stuart Russell, Peter Norvig	978-0136042594
Dracula	Bram Stoker	978-0141439846

1. Hash the books in the above table into a `HashSet` of size 5. In case of collisions, use collision lists. Insert the books in the same order as they appear in the table. What are the benefits and drawbacks of using collision lists?

2. Now hash the books into a `HashSet` of size 10 using linear probing. Insert the books in the same order as they appear in the table. What are the benefits and drawbacks of using linear probing? (*Hint:* The function for linear probing is $h(x, i) = (h'(x) + i) \mod m$)

3. Now hash the books into a `HashSet` of size 10 using quadratic probing. Insert the books in the reverse order as they appear in the table. What are the benefits and drawbacks of using quadratic probing? (*Hint:* The function for quadratic probing is $h(x, i) = (h'(x) + \frac{i(i+1)}{2}) \mod m$)

Exercise 11: Code Generation

Generate MIPS code for the following C0 programs. For that, draw an inference tree using the code generation derivation rules and give the MIPS code in the end. Use this offset function: $\mathit{offs} = \{x \mapsto 0, y \mapsto 4\}$.

1. `x = x + 2;`

2. `while (y) x = x + 2;`

You can reuse parts from previous exercises. If there are only small changes, mark them clearly in the derivation tree and the generated code.

Exercise 12:

1. Carefully observe the following program and identify its functionality. Provide a specification, including the precondition and postcondition, followed by a formal proof.

```
{ x = x + y;
  y = x - y;
  x = x - y; }
```

2. Take a look at the following program, find an invariant I and verify that this program is correct with respect to the specification (P, Q)

```
while ((x + 1) * (x + 1) <=  y)
{
  x = x + 1;
}
```

$$P := x = 0 \land y > 0 \quad Q := x * x \leq y < (x+1) * (x+1)$$