SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Reinhard Wilhelm
Jun.-Prof. Dr. Sebastian Hack
Jörg Herter, M.Sc.
Dipl.-Inform. Christoph Mallon

Compiler Construction WS09/10

Exercise Sheet 6

Please hand in the solutions to the theoretical exercises until the beginning of the lecture next Wednesday 2009-12-09, 10:00. Please write the number of your tutorial group or the name of your tutor on the first sheet of your solution. Solutions submitted later will not be accepted.

Exercise 6.1: Extreme Pointer (Points: 4)

We have seen in the lecture that stack and heap grow towards each other in the store S. SP and NP denote the Stack and the New pointer, respectively. During execution of a program we must make sure that SP and NP do not pass each other. Without any extra provisions we would have to compare SP and NP at every update of one of the two values. To save some work we introduce EP, the Extreme Pointer, which denotes the uppermost cell, to which SP may point to during the execution of the current function. This relieves us from checking for a collision whenever SP is manipulated. EP can be determined statically. It depends on the maximal stack usage during expression evaluation. Let $t(e)$ denote the number of stack cells needed to evaluate expression e. Assume e to be of the following form:

$$e ::= x \mid e_1[e_2] \mid e_1 = e_2 \mid e_1\ op_b\ e_2 \mid op_u e_1$$

Give a recursive definition for the computation of $t(e)$! Explain your definition briefly.

Exercise 6.2: Switch (Points: 2+2+2+2)

Translate the following switch statement into valid CMa code using the algorithm presented in the lecture. Use a context with $\rho(n) = (L, 3)$ and $\rho(i) = (G, 4)$.

```
switch (n)
{
    case 0: i = n+2; break;
    case 1: i = -n; break;
    case 2: i = 1; break;
    default: break;
}
```

Consider the following questions regarding switch statements.

1. How should the code function handle switch statements where gaps between case statements exist (i.e. cases are undefined)?
2. Is it always *feasible* to use jump tables to implement switch statements? Explain your answer!
3. Give a different alternative to implement code s for a general switch statement s. Discuss the (dis)advantages of your scheme.

Exercise 6.3: CMa (Points: 5)

Translate the following C program to CMa code using the rules presented in the lecture.

```
struct list
{
    int value;
    struct list *next;
};

struct list * create_list(int n)
{
    struct list *head;
    head = malloc(2);
    head->value = n;
    head->next = 0;
    if (n > 1)
    {
        struct list *tail;
        tail = create_list(n-1);
        head->next = tail;
    }
    return head;
}

int main(void)
{
    create_list(5);
    return 0;
}
```