1 Even More Code Generation 6+4 Points

- Translate the following C code into CMa code using the algorithm presented in the lecture. Remember to correctly compute the EP.

```
int sum;

int summarize (int n) {
    int i, sum;
    for (i = 0; i < n; ++i)
        sum = i+sum;
    return sum;
}

void main () {
    sum = summarize(42);
    return;
}
```

- Determine the state of the CMa before the execution of line 5 and after the execution of line 11. In each case, label variables in the stack and mark the stack frames.

2 Switch Statements 5+2+1+2 Points

- 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 $\rho_i = (L, 4)$.

```
int i;

switch (n) {
    case 0: i = n*(n+1); break;
    case 1: i = n; break;
    case 2: i = 1; break;
    default: break;
}
```
• How should the code function handle switch statements, where gaps between two case statements exist (e.g., line 4 missing in the above example)?

• Is it always feasible to use jump tables to implement switch statements? In case not, provide an example demonstrating your claim.

• Think of a different alternative to implement code $s$ $\rho$ for any switch statement $s$.

### 3 Heap 10 Points

Given the following structure definition:

```c
struct elem {
    int entry;
    struct elem *next;
};
```

Translate the following function into valid CMa code using the algorithm presented in the lecture. Remember to correctly compute the $EP$.

In this context, `list` is a variable of the type `struct elem *` with $\rho$ `list = (G, 1)`.

```c
void initialize (int n) {
    int i;
    struct elem *current = NULL;

    for (i = 0; i < n; ++i) {
        struct elem *temp = malloc(sizeof(struct elem));
        temp->entry = i;
        temp->next = NULL;

        if (!i)
            list = temp;
        else
            current->next = temp;

        current = temp;
    }
}
```