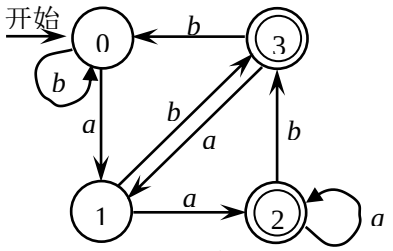
中 国 科 学 技 术 大 学

2009－2010 学年第二学期考试试卷（A）

考试科目：编译原理和技术　　得分：_____

学生所在系：_____ 姓名：_____ 学号：_____

1、（15 分）从教材第 2 章的习题中找出一个正规式，它所表示的语言就是右边的 DFA 所接受的语言。并请依据教材上提供的方法确认该 DFA 是最简的 DFA。



2、（10 分）下面是类型表达式的语法：

type → **integer** | **boolean** | **array**[**num**] **of** *type* | **record** *field_list* **end** | ↑ *type*

field_list → **id** : *type* | **id** : *type* ; *field_list*

若规定：在记录类型中不能出现数组类型（包括不能出现数组的指针类型）。请重新设计一个文法，把该约束体现在文法中，即它和上述文法的区别就是所定义的语言满足这个约束。

3、（10 分）根据教材 3.3.3 节的递归下降预测分析方法以及图 3.8 的方式，给第 2 题题目（而不是你答案）中的非终结符 *field_list* 写一个递归过程（若需要变换文法，则先变换文法）。

4、（10 分）为第 2 题题目的文法写一个翻译方案，若类型表达式不满足该题所规定的约束则报告错误。注：第 2 题通过重新设计文法，本题通过静态检查，来达到同样的目的。

5、（10 分）

（1）Java 语言的编译器通常把数组分配在堆上。Java 数组一般不能静态确定大小应该不是将它分配在堆上的原因，因为教材上图 6.12 给出了一种将不能静态确定大小的数组动态地分配在活动记录栈上的方法。Java 数组一般不能分配在活动记录栈上的原因是什么？

（2）将数组分配在活动记录栈上和分配在堆上给程序运行带来什么区别？

6、（5 分）有人认为，下面 C 程序中结构体类型 record 的定义方式可用来动态生成其中 a 数组的大小不一样的结构体，以适应某些编程场合的需要。你认为这样的程序能够通过 C 编译器的类型检查吗？请说明理由。

```
#include <malloc.h>
typedef struct {double r; int n; float a[];} record;
main() {
    record * p;
    p = malloc(sizeof(record) + sizeof(float) * 5);
    p->n = 5; p->a[4] = 100.0; …
}
```

7、（15 分）对于教材上 P_{172} 例 6.4 的程序（见下面 C 代码），GCC 编译器（版本：(GNU) 4.2.3

(Debian 4.2.3-5)）所生成的优化代码（见下面中间的汇编代码）同书上给出的代码（见下面左边的汇编代码）略有不同。请指出该较新版本编译器完成的优化（或采用的不同代码生成策略）。给一点提示：若把调用语句改成 func(j, j)，所生成的优化代码和下面中间的汇编代码的区别见下面右边的说明。

```
        func(i) long i; {
            long j;     j=i-1;      func(j);
        }
```

```
func:                          |   func:                          |
    pushl   %ebp               |       pushl   %ebp               |
    movl    %esp, %ebp         |       movl    %esp, %ebp         |
    subl    $4, %esp           |       subl    $4, %esp           | 该指令改成 subl $8, %esp
    movl    8(%ebp), %edx      |       movl    8(%ebp), %eax      |
    decl    %edx               |       subl    $1, %eax           |  在该指令和下条指令间
    movl    %edx, -4(%ebp)     |       movl    %eax, (%esp)       | 增加 movl %eax, 4(%esp)
    movl    -4(%ebp), %eax     |       call    func               |
    pushl   %eax               |       leave                      |
    call    func               |       ret                        |
    addl    $4, %esp           |
.L1:                           |
    leave                      |
    ret                        |
```

8、（5 分）请按教材上图 12.4 的方式给出第 375 页中类 Point 的对象表示。

9、（10 分）下面左右两边是一个 C 程序的两个源文件的内容：

```
#include <stdio.h>                  |   #include <stdio.h>
char *p = “0123456789”;             |   char p[10];
f () {                              |   main () {
    printf (“%s\n”, p);             |   f ();   p[1] = ‘1’;    f ();
}                                   |   }
```

运行目标程序所得到的结果如下（编译器是 GCC: (GNU) 4.2.3 (Debian 4.2.3-5) ）：

```
0123456789
Segmentation fault
```

请描述该目标程序各数据区的内容，并回答为什么运行时会出现 Segmentation fault。

10、（10 分）下面的 C 程序分别经非优化编译和 2 级以上（含 2 级）的优化编译后，生成的两个目标程序运行时的表现不同（编译器是 GCC: (GNU) 4.2.3 (Debian 4.2.3-5)）。请回答它们运行时的表现有何不同，并说明原因。

```
int f(int g( ) ) {
    return g(g);
}
main() {
    f(f);
}
```