Klausur zu Allgemeine Informatik III
14. Februar 2007 (WS 2006/2007)

Bearbeitungszeit: 120 Minuten
NICHT MIT BLEISTIFT SCHREIBEN!

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Nr	Max	Bewertung	
1	**13**	xxxxx	
(a)	3		xxxxx
(b)	3		xxxxx
(c)	3		xxxxx
(d)	4		xxxxx
2	**8**	xxxxx	
3	**8**	xxxxx	
(a)	2		xxxxx
(b)	3		xxxxx
(c)	3		xxxxx
4	**8**	xxxxx	
(a)	6		xxxxx
(b)	2		xxxxx

Nr	Max	Bewertung	
5	**12**	xxxxx	
(a)	6		xxxxx
(b)	6		xxxxx
6	**10**	xxxxx	
7	**8**	xxxxx	
(a)	2		xxxxx
(b)	4		xxxxx
(c)	2		xxxxx
8	**9**	xxxxx	
(a)	3		xxxxx
(b)	3		xxxxx
(c)	3		xxxxx
9	**14**	xxxxx	
Summe	**90**		

Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe oder die Rückseite unter Angabe der Aufgabe. Nennen Sie möglichst alle Annahmen, die Sie für die Lösung einer Aufgabe treffen!

Viel Erfolg!

Aufgabe 1 **(13 Punkte)**

Gegeben seien die folgenden Definitionen:

```
int x = 2, y = 3, a = 4, b = 5;
#define NOTMAX(x,y) x>y ? x : y
```

Geben Sie für die folgenden Fälle jeweils an, wie der Präprozessor das Makro `NOTMAX` expandiert und welcher Wert sich für den Ausdruck zur Laufzeit ergibt. **Lesen Sie genau!**

(a) 3 Punkte

```
NOTMAX(a,3);

a>3 ? a : 3;     ==> 4
```

(b) 3 Punkte

```
NOTMAX(y,x);

y>x ? y : x;     ==> 3
```

(c) 3 Punkte

```
NOTMAX(++x,1);

++x>1 ? ++x : 1;     ==> 4
```

(d) 4 Punkte

```
NOTMAX(b,NOTMAX(6,7));

b>6>7 ? 6 : 7 ? b : 6>7 ? 6 : 7;      ==> 5;
```

Aufgabe 2 **(8 Punkte)**

Gegeben sei das folgende Programm:

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

int obscure (int * x, int * y, int z)
{
        if (x == y) {
                return 1;
        }
        (*x)++;
        if (*x == *y) {
                return 2;
        }
        if (*x == z) {
                return 3;
        }
        if (*y == z) {
                return 4;
        }
        return 5;
}

int main ()
{
        int a = 1, b = 2, ret;
        ret = obscure (&a, &b, 1); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&a, &a, 2); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&b, &a, 3); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&a, &b, 4); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&a, &b, a); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&a, &b, b); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&b, &a, 2); printf ("%d %d %d\n", a, b, ret);
        ret = obscure (&b, &a, 1); printf ("%d %d %d\n", a, b, ret);
        return 0;
}
```

Welche Ausgabe erzeugt dieses Programm?

```
2 2 2
2 2 1
```

```
2  3  3
3  3  2
4  3  4
5  3  4
5  4  5
5  5  2
```

Aufgabe 3 **(8 Punkte)**

(a) 2 Punkte

Geben Sie **eine** Deklaration an, mit der zwei Variablen p und q vom Typ „Zeiger auf Integer“ deklariert werden.

```
int * p, * q;
```

(b) 3 Punkte

Beschreiben Sie verbal, was hier genau deklariert wird:

```
int (*X) (double);
```

`X` ist ein Zeiger auf eine Funktion, die `int` zurückliefert und als Argument einen Parameter vom Typ `double` bekommt.

(c) 3 Punkte

Geben Sie eine Deklaration für ein zehnelementiges Array mit dem Namen X an. Die Elemente des Arrays sollen Zeiger auf Funktionen sein, die `void` zurückliefern und keine Parameter haben.

```
void (*X[10])(void)
```

Aufgabe 4 **(8 Punkte)**

Ein Palindrom ist eine Zeichenfolge, die sowohl vorwärts als auch rückwärts gelesen den gleichen Text ergibt.

(a) 6 Punkte

Ergänzen Sie die Prozedur `make_palindrome` so, daß `str` anschließend ein Palindrom enthält. Dabei soll der bereits vorhandene Anfang unverändert bleiben, es werden also nur Zeichen angehängt.
Der Rückgabewert ist 1, falls alles gut ging und 0 falls ein Fehler auftrat, z.B. weil zu wenig Speicher vorhanden war. Der Text in `str` soll im Fehlerfall nicht verändert werden.

```
int make_palindrome (struct stralloc * str)
{
        int orig_len = str->len;
        int i = str->len-1;
        while (--i >= 0) {
                if (!stralloc_catb (str, str->s+i, 1)) {
                        str->len = orig_len;
                        return 0;
                }
        }
        return 1;
}
```

(b) 2 Punkte

Wie gehen Sie vor, um nach einem erfolgreichen Aufruf von `make_palindrome` das Palindrom auf die Standardausgabe auszugeben?

```
stralloc_0 (&str);
printf ("%s\n", str.s);
```

Aufgabe 5 **(12 Punkte)**

Gegeben sei die folgende Datenstruktur für die Verwaltung einer einfach verketteten Liste. Dabei wird eine leere Liste durch den Nullzeiger repräsentiert.

```
struct list {
        struct list * next;
        char * key;
        int val;
};
```

Eine solche Liste kann dazu verwendet werden, um Strings Integerzahlen zuzuordnen. D.h. jedes Element der Liste enthält im Feld `key` einen Zeiger auf einen (in der gesamten Liste eindeutigen) String und im Feld `val` den Integerwert, der diesem String zugeordnet wird.
In den folgenden beiden Teilaufgaben werden zwei Funktionen implementiert, die eine solche Liste manipulieren. Zum besseren Verständnis dieser Funktionen ist hier noch ein kurzes Hauptprogramm abgedruckt, zu dem die beiden Funktionen passen sollen.

```
int main()
{
        char text[4] = "AAA";
        struct list  * l = 0;
        set (&l, text, 10);
        text[0] = 'B'; text[1] = 'B'; text[2] = 'B';
        set (&l, text, 20);
        set (&l, "AAA", 30);
        printf ("%d\n", get_and_remove (&l, "AAA"));   /* 30 */
        printf ("%d\n", get_and_remove (&l, "BBB"));   /* 20 */
        printf ("%d\n", get_and_remove (&l, "CCC"));   /*  0 */
        return 0;
}
```

(a) 6 Punkte

Ergänzen Sie die folgende Prozedur `set`, so daß anschließend dem String `str` der Wert `newval` zugeordnet wird. Sie können davon ausgehen, daß vor dem Aufruf höchstens ein Element in der Liste existiert, bei dem der String im Feld `key` mit dem Parameter `str` übereinstimmt. Das soll natürlich auch nach dem Aufruf von `set` noch der Fall sein. **Speicherallokation nicht vergessen!**

```
void set (struct list ** listptr, char * str, int newval)
{
        struct list * p = (*listptr);
        while (p) {
                if (strcmp (str, p->key) == 0) {
                        p->val = newval;
                        return;
                }
                p = p->next;
        }
        p = malloc (sizeof (struct list)); assert (p);
        p->key = strdup (str); assert (p->key);
        p->val = newval;
        p->next = (*listptr);
        (*listptr) = p;
}
```

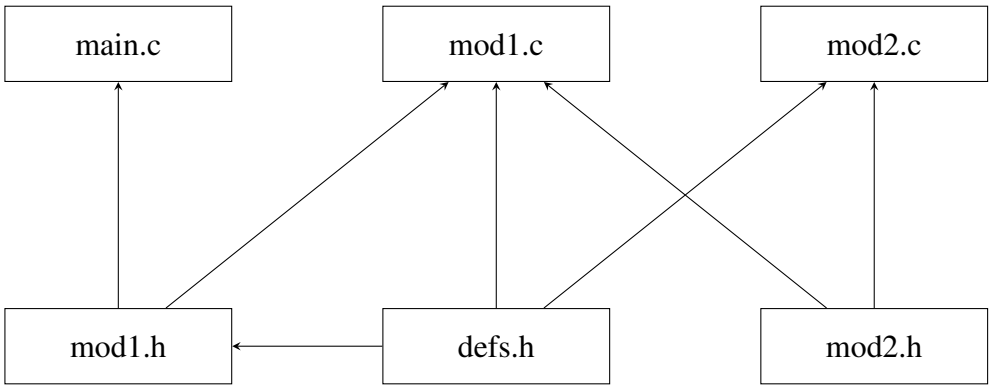
(b) 6 Punkte

Schreiben Sie jetzt eine Prozedur `get_and_remove`, die zu einem gegebenen String `str` den Wert, der zu diesem String gehört, zurückliefert und das Element aus der Liste entfernt. Wenn kein passendes Element vorkommt, soll Null zurückgeliefert werden. **Achten Sie darauf, daß eventuell allokierter Speicher wieder freigegeben wird.**

```
int get_and_remove (struct list ** listptr, char * str)
{
        struct list ** pp = listptr;
        struct list * todel;
        int ret;
        while (*pp) {
                if (strcmp ((*pp)->key, str) == 0) {
                        todel = *pp;
                        (*pp) = todel->next;
                        ret = todel->val;
                        free (todel->key);
                        free (todel);
                        return ret;
                }
                pp = &(*pp)->next;
        }
        return 0;
}
```

Aufgabe 6 **(10 Punkte)**

Gegeben sei ein Programm, das wie im folgenden Bild dargestellt aufgebaut ist. Dabei stellt jedes Rechteck eine Quelldatei dar. Ein Pfeil gibt an, daß die Datei am Anfang des Pfeils von der Datei am Ende des Pfeils mittels `#include` eingebunden wird.



Zusätzlich verwendet die Datei `mod1.c` die Mathematik-Bibliothek und die Datei `mod2.c` verwendet die `stralloc`-Bibliothek.
Schreiben Sie ein Makefile, das aus diesen Dateien ein ausführbares Programm mit dem Namen `main` erzeugt. Dabei müssen die angegebenen Abhängigkeiten und Rahmenbedingungen beachtet werden und es sollten nur die Dinge neu übersetzt werden, bei denen dies wirklich notwendig ist.

```
.PHONY: clean all
all:    main
clean:
        rm -f *.o main core a.out
main:   main.o mod1.o mod2.o
        gcc -Wall -omain -L /usr/local/diet/lib main.o mod1.o mod2.o -low
main.o: main.c mod1.h defs.h
        gcc -c -Wall main.c
mod1.o: mod1.c mod1.h defs.h mod2.h
        gcc -c -Wall mod1.c
mod2.o: mod2.c mod2.h defs.h
        gcc -c -Wall -I /usr/local/diet/include mod2.c
```

Aufgabe 7 **(8 Punkte)**

(a) 2 Punkte
Beschreiben Sie ganz allgemein, welche Aufgaben ein Dateisystem hat. `Verwaltung von Daten auf Massenspeichermedien, Abtraktion der Daten in Form von Dateien`

(b) 4 Punkte
Welche beiden Datenstrukturen **auf der Platte** spielen im Zusammenhang mit der Verwaltung eines Dateisystems eine große Rolle? Beschreiben Sie jeweils kurz die Aufgabe dieser beiden Datenstrukturen.

Inode Verwaltungsinformationen für eine Datei (Größe, Lage der Datenblöcke, Rechte, etc.)

Superblock Verwaltungsinformationen für das gesamte Dateisystem.

(c) 2 Punkte
Welche zwei Bedingungen müssen erfüllt sein, damit der zu einer Datei gehörende Platz auf der Platte wieder freigegeben werden kann?

Link counter ist Null Es darf keinen Verzeichniseintrag mehr geben, der auf die Datei verweist.

Kein Filepointer mehr Es darf keinen Prozeß geben, der die Datei noch geöffnet hält.

Aufgabe 8 **(9 Punkte)**

(a) 3 Punkte

Sie erfahren, dass es unter Solaris einen Systemaufruf *wait* gibt. Wie sollten Sie vorgehen, um mehr Informationen, über diesen Systemaufruf zu erfahren? Wo genau können Sie in Erfahrung bringen, welche Header-Dateien eingebunden werden müssen?

```
man -s2 wait
```

Die notwendigen Headerdateien findet man im Abschnitt SYNOPSIS der Manualseit vor dem Prototyp der Funktion.

(b) 3 Punkte

Sie wollen eine Datei mit dem Namen `/tmp/hallo/welt` erzeugen. Ob die Datei bereits existiert, ist dabei egal. Allerdings soll sichergestellt sein, daß der Inhalt der Datei am Ende nur der Text `Hallo Welt!` enthält, egal ob die Datei vorher bereits existiert hat oder nicht. Wenn die Datei neu angelegt wird, soll nur der Aufrufer Lese- bzw. Schreibrechte auf der Datei haben. Geben Sie die dazu notwendigen Systemaufruf (mit allen Parametern) an. Auf die eigentlich notwendige Fehlerbehandlung kann hier verzichtet werden.

```
int fd = open ("/tmp/hallo/welt",
               O_WRONLY | O_CREAT | O_TRUNC, 0666);
write (fd, "Hallo Welt!\n", 12);
close (fd);
```

(c) 3 Punkte

Nennen Sie mindestens drei plausible Gründe, warum die in Teilaufgabe (b) von Ihnen verwendeten Systemaufrufe fehlschlagen können. Geben Sie dabei jeweils an, auf welchen Systemaufruf sich die Begründung bezieht.

- open: `/tmp/hallo/` existiert nicht bei open.
- open: Die notwendigen Rechte zum Anlegen bzw. öffnen der Datei fehlen.
- write: Es ist kein Platz mehr auf der Platte.

Aufgabe 9 **(14 Punkte)**

Schreiben Sie ein Programm, das *im aktuellen Verzeichnis* den *Namen* und die *Größe* der *regulären* Datei ausgibt, auf die *zuletzt zugegriffen* wurde.
Hinweis: Die Zeit des letzten Zugriffs findet man in der relevanten Datenstruktur im Feld `st_atime`. Sie wird in Sekunden seit dem 1. Jan 1970 gemessen.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <dirent.h>
#include <stralloc.h>

int main ()
{
        DIR * dir;
        struct dirent * dent;
        stralloc best = {0};
        size_t best_size = 0;
        time_t best_access = 0;
        struct stat statbuf;

        dir = opendir (".");
        if (dir == NULL) {
                perror ("opendir");
                exit (1);
        }
        errno = 0;
        while (1) {
                errno = 0;
                dent = readdir (dir);
                if (errno) {
                        perror ("readdir");
                        exit (1);
                }
                if (dent == NULL) {
                        break;
                }
                if (stat (dent->d_name, &statbuf) < 0) {
                        perror ("stat");
                        exit (1);
```

```
                }
                if ((statbuf.st_mode & S_IFMT) != S_IFREG) {
                        continue;
                }
                if (statbuf.st_atime > best_access) {
                        best_access = statbuf.st_atime;
                        best_size = statbuf.st_size;
                        stralloc_copys (&best, dent->d_name);
                }
        }
        closedir (dir);
        if (best.len) {
                stralloc_0 (&best);
                printf ("%s %d\n", best.s, best_size);
        }
        return 0;
}
```