Aufgabe 1: Was gibt das im folgenden angegebene Programm bei seiner Ausfuehrung auf den Bildschirm aus?

```
#include <stdio.h>

typedef struct {
         int *iptr1, *iptr2, *iptr3;
      } S3IPTR;

int main(void)
{
   S3IPTR s3iptr, *ps3iptr;
   int i=1, j=2, k=3, l, *pint;

   ps3iptr = &s3iptr;
   s3iptr.iptr1 = &i;
   s3iptr.iptr2 = &k;
   s3iptr.iptr3 = &j;
   pint=s3iptr.iptr2;
   l = *s3iptr.iptr1+10;
   *(*ps3iptr).iptr1 = k;
   ps3iptr->iptr2 = s3iptr.iptr3;
   ps3iptr->iptr3 = pint;
   *s3iptr.iptr3 = l;

   printf("%d\n", i);  /* i=_____ */
   printf("%d\n", j);  /* j=_____ */
   printf("%d\n", k);  /* k=_____ */

   printf("%d\n", *s3iptr.iptr3); /* *s3iptr.iptr3=_____ */
   printf("%d\n", *(*ps3iptr).iptr2);
                          /* *(*ps3iptr).iptr2=_____ */
   printf("%d\n", *ps3iptr->iptr1);
                            /* *ps3iptr->iptr1=_____ */
   return(0);
}
```

Aufgabe 2: Was gibt das im folgenden angegebene Programm bei seiner Ausfuehrung auf den Bildschirm aus, wenn es mit den Kommandozeilenparametern 7900315 und 7948265 aufgerufen wird?

```
#include <stdio.h>

int main(int argc, char *argv[])
{
   char *s;
   char code[] = "SKIAMXFNLE";
   s = *++argv;
   s += argc+2;
   while (*s - '7')
   {
      printf("%c", code[*s-- - '0']);
   }
   return 0;
}
```

Aufgabe 3: Geben Sie die Implementierung einer C-Funktion str2lower an, die alle Grossbuchstaben eines uebergebenen C-Strings in Kleinbuchstaben umwandelt und die Anfangsadresse des C-Strings als Ergebnis zurueckliefert. Umlaute sollen dabei nicht beruecksichtigt werden.

Aufgabe 4: Spezifizieren Sie die Funktion LoescheListe in C in rekursiver und nicht rekursiver Form.

LoescheListe erhaelt als Parameter einen Zeiger auf eine verkettete Liste. Ist die Liste nicht leer, werden alle Listenelemente der Liste geloescht, d.h. der durch sie belegte Speicherplatz wird freigegeben, der als "Referenzparameter" uebergebene Zeiger auf den Listenanfang wird auf NULL gesetzt und als Funktionsergebnis wird TRUE zurueckgeliefert. Ist die Liste leer, so ist das Funktionsergebnis FALSE.

Bei der Spezifikation der Funktion ist von folgendem Aufbau eines Listenelementes und den Datentypen seiner Komponenten auszugehen:

```
typedef struct Listenelement {
                    int Datenteil;
                    struct Listenelement *Naechstes;
                } LISTENELEMENT;
```

Aufgabe 5: Angenommen in einem binaeren Baum ist in jedem Knoten ein int-Wert unter dem Komponentennamen IntWert abgespeichert. Spezifizieren Sie die Funktion KnotenSumme, welche die int-Werte aller Knoten zu einer Summe vom Typ double aufsummiert und als Funktionsergebnis zurueckliefert.

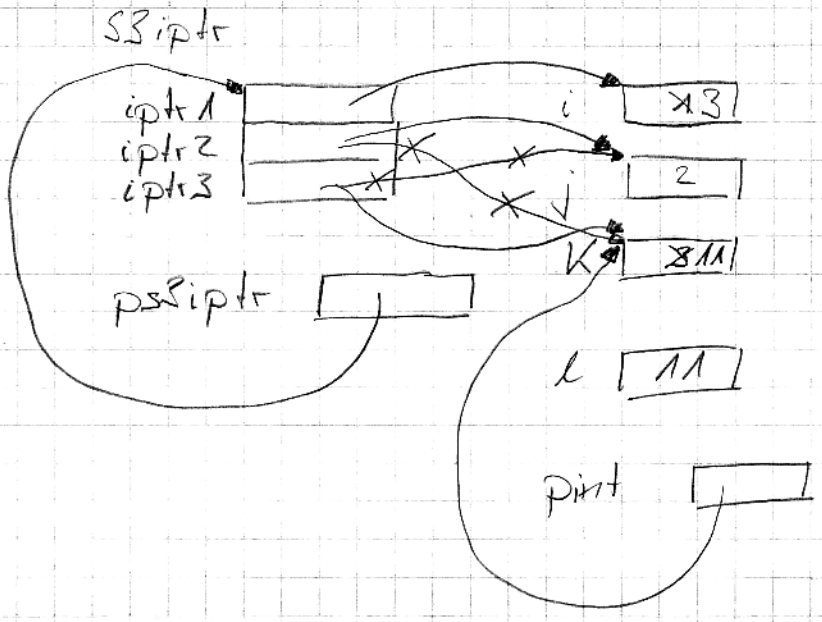
Bei der Spezifikation der Funktion ist von folgendem Aufbau eines Baumknotenelementes und den Datentypen seiner Komponenten auszugehen:

```
typedef struct Baumknotenelement {
                    int IntWert;
                    struct Baumknotenelement *Linker;
                    struct Baumknotenelement *Rechter;
                } BAUMKNOTENELEMENT;
```

Aufgabe 6: Spezifizieren Sie das C-Programm CP2END. CP2END erhaelt ueber die Kommandozeile zwei Dateinamen uebergeben und kopiert, falls die zweite Datei existiert und der erste Dateiname ein gueltiger Dateiname ist, den Inhalt der zweiten Datei an das Ende der ersten Datei. Falls eine Datei mit dem ersten Dateinamen noch nicht existiert, erstellt CP2END unter diesem Namen eine Kopie der zweiten Datei.

Informatik - Probeklausur II

Aufgabe 1

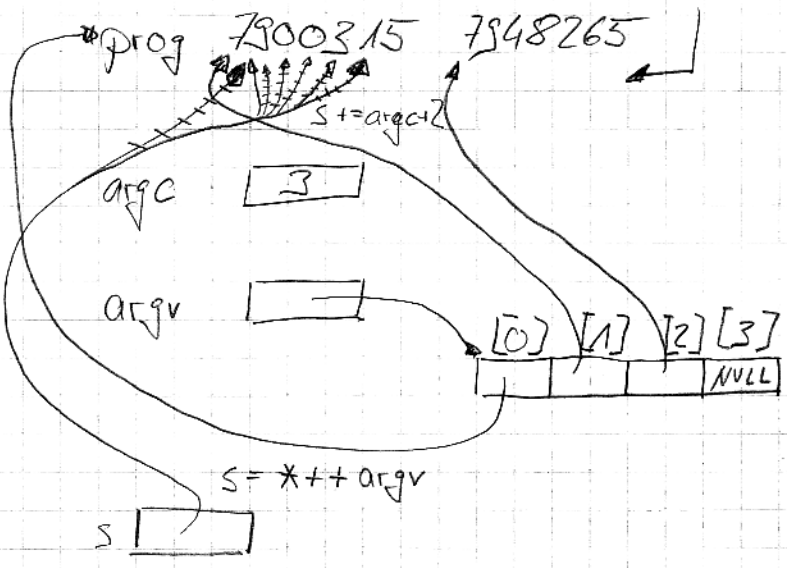


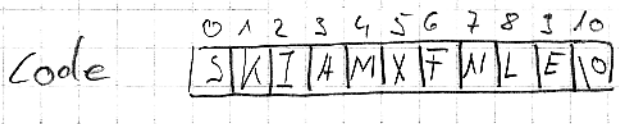
$i = 3 \qquad j = 2 \qquad k = 11$

$^{*}sS3iptr.iptr3 = 11 \qquad ^{*}(^{*}ps3iptr).iptr2 = 2$

$^{*}ps3iptr \rightarrow iptr1 = 3$

Aufgabe 2





KASSE

Aufgabe 3

```
char *str2lower (char *s)
{
   char *t = s;
   for ( ; *s; s++)
   { if (isupper ((int)*s))
        *s = (char) (tolower ((int)*s));
   }
   return t;
}
```

Aufgabe 4

Rekursive Form:

```
int LoescheListe (LE **ppListe)
{
   if (*ppListe != NULL)
   {
      LoescheListe (R (**ppListe)_Naechstes);
      free (*ppListe);
      *ppListe = NULL;
      return TRUE;
   }
   else
      return FALSE;
}
```

Nicht - Rekursive Form:

```
Bool Loesche Liste ( LE ** root)
{
    if (*root == NULL)
        return FALSE;
    LE *act.LE  = * root;
    LE *next.LE = act.EL -> Naechstes;
    while (act.EL != NULL)
    {
        free (act.EL);
        if (next.EL == 0)
            break;
        else
            act.EL = next.EL;
            next.EL = actEL -> Naechstes;
    }
    *root = NULL;
    return TRUE;
}
```

Aufgabe 5:

```
double KnotenSumme (BKE *pWurzel)
{
  if (pWurzel != NULL)
    return (*pWurzel).intWert + KnotenSumme ((*pWurzel).Links
                              + KnotenSumme ((*pWurzel).Rechts
  else
    return 0;
}
```

A