

KSP - Tutorium

Test 4 Africa's Geek



KSP

Test für die Klausurvorbereitung – KSP

| | |
|-----------|----------------------|
| Nachname: | |
| Vorname: | |
| Betreuer: | Donchi Fofack Donald |

| | Max. Punktzahl | erreicht |
|-----------|----------------|----------|
| Aufgabe 1 | 17 | |
| Aufgabe 2 | 20 | |
| Aufgabe 3 | 20 | |
| Aufgabe 4 | 10 | |
| Aufgabe 5 | 35 | |
| Gesamt | 102 | |

Aufgabe 1: (17 Punkte)

I- Schreiben Sie C-Deklarationen für folgende Aussagen:

- a) eine möglichst große ganze Zahl z
- b) ein Feld a von 20 Zeichen
- c) ein Feld b von 10 Zeigern auf ganze Zahlen
- d) eine Funktion f , die ein Feld von ganzen Zahlen ohne Vorzeichen und dessen Größe erwartet und einen Zeiger auf ein Element dieses Feldes zurückgibt
- e) ein "Byte" (vorzeichenloses Zeichen) b
- f) ein Feld a von 4 Zeigern auf vorzeichenlose ganze Zahlen
- g) eine Funktion f , die zwei ganze Zahlen erwartet, und einen Zeiger auf eine ganze Zahl zurückgibt.
- h) einen Zeiger p auf eine Funktion, die einen Zeiger auf ein Zeichen erwartet und einen Zeiger auf ein Zeichen zurückgibt.

II- Es sei vorgegeben:

```
int a[] = {1, 3, 5, 7, 2, 4, 6, 8};
```

```
int *p = &a[4];
```

```
int **q = &p;
```

```
int b[] = {10, 8, 15, 9, 1, 2, 12};
```

```
int *k = &b[3];
```

Geben Sie die Werte der folgenden Ausdrücke an:

- a) $a[3] + a[4]$
- b) $*(p - 2)$
- c) $*p + 2$
- d) $((*q - 1) - 3)$
- e) $b[5]$
- f) $b[b[6] - b[0]]$
- g) $*k - 1$

h) $*(k-1)$

I) $\&b[6] - k$

Aufgabe 2: (20 Punkte)

I-

Es sollen ganze Zahlen in einer doppelt verketteten Liste abgelegt werden. Zur Erinnerung: ein Knoten einer doppelt verketteten Liste hält jeweils einen Zeiger auf seinen Vorgänger und auf seinen Nachfolger (und speichert hier eine ganze Zahl).

a) Definieren Sie einen Typ "Node" zum Speichern eines Knotens einer solchen Liste.

b) Definieren Sie einen Zeiger p, der auf eine solche Liste zeigen kann.

II-

Gegeben ist folgende Datenstruktur einer doppelt verketteten Liste:

```
typedef struct element {
    struct element * left;
    struct element * right;
    int value;
} ListElement;
```

left muss dabei immer auf das Element links in der Liste vom aktuell betrachteten Element zeigen (oder beim ersten Element auf NULL).

right muss immer auf das Element rechts neben dem aktuell betrachteten Element in der Liste zeigen (oder auf NULL beim letzten Element).

- a. Implementieren Sie folgende Funktion, die einen Wert in eine Liste einfügen soll. Der einzufügende Wert value soll anschließend an n-ter Stelle in der Liste stehen. Der Rest der Liste soll erhalten bleiben, wird also um eine Stelle nach rechts verlegt. Achten Sie darauf jedes neue Listen Element auf dem Heap anzulegen.

```
void insert_value(ListElement* list, int value, int n)
{
```

Aufgabe 3: Wissensfragen(20 Punkte)

A- Betrachten Sie die Bigint-Bibliothek unter Berücksichtigung des vorgegebenen Garbage Collection Verfahrens der Ninja VM. Mit welchen Problemen hätte eine Implementierung der Garbage Collection zu tun, würde die Schnittstelle der Bigint Bibliothek, anstelle der Big Integer Processor (bip) Variable, Übergabeparameter und Rückgabewerte vom Typ ObjRef für die Funktionsaufrufe definieren?

Beispielhaft gezeigt also: ObjRef result = bigAdd(a, b);
Beschreiben Sie ausführlich.

B- Beschreiben Sie die drei Phasen des Stop & Copy Garbage Collection Verfahrens.

C- Welche Aufgabe hat der Stack Pointer?

D- Wozu dient in C der tag bei einem Struct oder einer Union?

Aufgabe 4: (10 Punkte)

Programmierer Schlaufuchs muss häufig das Produkt aus zwei Faktoren berechnen. Er definiert dazu einen Makro:

```
#define MUL(a,b) a*b
```

a) Zu welchem Ausdruck wird der Aufruf MUL(2,3) expandiert und was ist dessen Wert?

b) Zu welchem Ausdruck wird der Aufruf MUL(5-3,1+2) expandiert und was ist dessen Wert?

c) Helfen Sie dem armen Schlaufuchs mit einer korrekten Definition des Makros aus der Patsche!

Aufgabe 5: (35 Punkte)

Schreiben Sie in Ninja Assembler Folgendes Programm:

A- Schreiben Sie eine Funktion mit dem Namen `print_n_times`, die zwei Zahlen als Parameter entgegen nimmt (`n` und `p`) und eine Zahl zurück gibt. (10 Punkte)

Die Funktion schreibt `n` Zeilen mit der Zahl `p` in die Standardausgabe. Der Rückgabewert entspricht der Summe beider Parameter. Im Fall, dass `n` negativ ist, wird nichts ausgegeben und der Rückgabewert ist `-1`.

Implementieren Sie außerdem einen Funktionsaufruf in Ninja Assembler. Dabei rufen Sie die Funktion mit den Parametern `n=3` und `p=25` auf und schreiben Sie den Rückgabewert in die Standardausgabe.

Die Ausgabe sollte also folgendermaßen aussehen:

25
25
25
28

B- schreiben Sie bitte `funk1`, `funk2`, `funk3`, `main` und `funk4` in assembler.

```
void funk1(){
    writeInteger(12);
    writeCharacter('\n');
}

void funk2(Integer a, Integer b){
    writeInteger(a*b);
    writeCharacter('\n');
}

Integer funk3(){
    return 2021;
}

Integer funk4(Integer a, Integer b){
    local Integer c;
    c = a / b;
    return c + 10;
}

void main(){
    local Integer a;
    local Integer b;
    funk1();
    funk2(1,2);
    a = funk3();
    b = funk4(1,2);
    writeInteger(b*a);
    writeCharacter('\n');
}
```