Name: .....
Note: ...

TD KSP

Aufgabe 1

1. Was ist ksp?
2. Was ist ein Compiler?
3. Was ist ein Assembler?
4. Was ist ein C-Präprozessor?
5. Wie nennt man Dateien mit Endung .h?
6. Was ist der Unterschied zwishen #include"..." und #include<...>?
7. Was ist ein Makros?
8. Was ist bedingte Compilierung?
9. Was ist VM?
10. Was ist ein Stack?
11. Was sind die Funktionalität eines Stacks?
12. Wie werden Operationen im Stack ausgeführt?
13. Wie ist ein Steuerwerk in c definiert?
14. Wie ist ein Pointer in C definiert?
15. Was ist eine lokale Variable in C?
16. wie funktioniert void * malloc(size_t size);
17. Wie wird aus einem Quellcode ein ausführbares Programm erstellt?
18. Gibt es die Möglichkeit in C für sein Programm Speicher anzufordern?
19. Welche Speicherbereiche gibt es in einem C-Programm und was kann darin enthalten sein? (Denken Sie an den Addressaufbau eines Programms)
20. Wozu dient in C der tag bei einem Struct oder einem Union?
21. Wie können Sie das Schlüßelwort static in c nutzen?
22. Was bewirkt das Schlüßel wort extern?(wofür wird es benötigt und worauf muss geachtet werden?)
23. Wozu dienen Header-Dateien?
24. Wozu muss eine .c datei ihre eigene Header-datei inkludieren?
25. Beschreiben Sie wie man Mehrfachinklusion von Header-dateien verhindert?
26. was ist die Aufgabe des Präprozessors?(wann wird er ausgeführt?)
27. Wie definieren Sie Makro für den Präprozessor?, Worauf müssen Sie bei der Definition von Makro achten?
28. Wie können Sie in c speicher nutzen, der sich auf dem Heap befindet?
29. Wie können Sie in c speicher nutzen, der sich auf dem Stack befindet?
30. Welche Aufgabe hat der Stack Pointer?
31. Welche Aufgabe hat der Frame Pointer?

32. Welche Aufgabe hat der Programm Counter?
33. Was ist ein Stackframe und wozu dient er?
34. Was bedeutet eine Kurzschlußauswertung bei Bedingungen und welche Alternative gibt es?
35. Was bedeutet vollständigauswertung bei Bedingungen und welche Alternative gibt es?
36. Beschreiben Sie die drei Phasen des Stop & Copy garbage collection verfahren.
37. Was bedeutet ein gesetztes Broken Heart Flag bei der Garbage Collection(Stop&Copy)?
38. Definieren Sie ein Makro: Set(x,n), welches das n-te Bit in der Zahl x setzt.
    (a) Definieren Sie das Makro.
    (b) Setzen Sie mit dem Makro das 7-te Bit in der Zahl 0x47.
39. Ergänzen Sie die umgangssprachlichen Erklärungen zu folgenden Deklarationen (benutzen Sie dabei deutsche Begriffe):

```
int a[25];
unsigned int b;
char* c;
void (*func) (int x, int y);
char* argv[];
```

40. Was kommt in eine .c Datei und was kommt in eine .h Datei?
41. Schreiben Sie in Ninja Assembler ein Programm, welches die Fakultät von 10 mit einer Schleife berechnet.
42. Schreiben Sie C-Deklarationen für folgende Objekte:
    (a) ein "Byte" (vorzeichenloses Zeichen)
    (b) ein Feld a von 4 Zeigern auf vorzeichenlose ganze Zahlen
    (c) eine Funktion f, die zwei ganze Zahlen erwartet, und einen Zeiger auf eine ganze Zahl zurickgibt
    (d) einen Zeiger p auf eine Funktion, die einen Zeiger auf ein Zeichen erwartet und einen Zeiger auf ein Zeichen zurückgibt
    (e) Schreiben Sie eine Funcktion in Ninja-Assembler, die ein Programm, welches die Fakultät einer belibigen Zahl berechnet.

Aufgabe 2

1. Schreiben Sie eine Datenstruktur, um einen binärbaum in C darzustellen. Hierbei sollen Knoten ein Zeichen und zwei weitere Knoten enthalten. Blätter sollen vorzeichenbehaftete Ganzzahlen enthalten.
2. Schreiben Sie eine Funktion in C, die zwei Knoten und ein Zeichen entgegennimmt, einen neuen Elternknoten für die beiden Knoten erstellt und einen Verweis auf diesen Elternknoten zurück gibt. Beachten Sie mögliche auftretende Fehler!

Aufgabe 3

1. Geben Sie für jede Komponente der u.a. Struktur an, wie viele Bytes dieser Datentyp belegt. (Schreiben Sie die Werte direkt hinter den Datentyp in den Code.) Tragen Sie zusätzlich in der Tabelle ein, wie viele Bytes die Struktur im Speicher minimal und tatsächlich belegt. (Nehmen Sie hierbei an, dass die Architektur des Zielsystems X86_64 ist.)

```
struct mystruct{
    void *p; //1
    int i; //2
    unsigned int u; //3
    long long l; //4
    char *str; //5
    char c[5]; //6
}mystruct;
```

- minimale Größe: ..... Bytes
- tatsächliche Größe: ..... Bytes

2. Gegeben seien die folgenden bereits definierten und initialisierten Variablen:

```
int x = 0x41;
char c = 0x42;
char *s = "Hallo";
char t[5] = {'W', 'e', 'l', 't', '\0'};
int *y = &x;
unsigned int u = -1;
```

Geben Sie die entsprechenden 6 printf()-Anweisungen an, die exakt die folgende Ausgabe erzeugen. Benutzen Sie dafür die o.a. Variablen und die jeweiligen Formatierungsanweisungen von printf():
Anmerkung: Die u.a. Speicheradressen stammen von einem realen Programmaufruf - lassen Sie sich hierdurch nicht irritieren! Verwenden Sie hierfür einfach die Formatierungsanweisung, die eine Architektur-abhängige Speicheradresse ausgibt!

```
0x7ffd62a39d48: 0x41 65 A//Beispiel: printf("%p: 0x%x %d %c\n", &x, x, x, x);
0x7ffd62a39d47: 0x42 66 B
0x7ffd62a39d50: 0x55df2e4fc004 Hallo
0x7ffd62a39d63: 0x7ffd62a39d63 Welt
0x7ffd62a39d58: 0x7ffd62a39d48 0x41 65 A
0x7ffd62a39d4c: 0xffffffff 4294967295 3261177920
```

3. Gegeben seien die 2 folgenden Header-Dateien debug.h und helper.h und die Datei main.c. Ändern Sie die Header-Dateien so ab, sodass keine Mehrfachinklusion und keine zyklische Inklusion auftreten kann. (Schreiben Sie Ihre Lösung direkt in den u.a. Codebereich.)

- debug.h

```
/** debug.h - start **/
.......
.......
void debug(int, char *);
.......
/** debug.h - end **/
```

- helper.h

```
/** helper.h - start **/
.......
.......
 #include "debug.h"

 int convert_to_int(char c);

.......
/** helper.h - end **/
```

- main.c

```
#include "helper.h"
#include "debug.h"

 int main(void){
    ...
 }
```

4. Gegeben seien die Dateien main.c, debug. c und helper.c. Wie lautet der Aufruf, um aus diesen 3 Source Code Dateien das lauffähige Programm mein_programm zu erstellen? Hierbei sollen folgende Optionen im Compiler/Linker gesetzt sein:
   - Es sollen alle Warnungen ausgegeben werden.
   - Es sollen Debug-Symbole im Programm vorhanden sein.
   - Das Programm soll nach den C99 Standard kompiliert werden.
   - Das Programm soll sich "pedantisch" an den Standard halten.
5. Definieren Sie 3 Makros, die folgende Funktionen umsetzen:
   - Ein Makro mit dem Namen FUNC1, welches die Funktion (a+b)/(a*b) umsetzt.
   - Ein Makro mit dem Namen HAS_MSB_SET, welches prüft, ob bei einer Variablen vom Datentyp long das sog. Most-Significant-Bit (MSB) gesetzt ist (hierzu dürfen Sie sich ein Hilfsmakro definieren).

Aufgabe 4

1. Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code(Integer c){
    local Integer a;
    local Integer resultat;
    a = c;
    resultat = 1;
    writeInteger(a);
    resultat = 1;
    writeInteger(a);
    writeCharacter('!');
    writeCharacter('=');
    while(a>0){
        resultat = resultat * a;
        a = a - 1;
    }
    writeInteger(resultat);
    writeCharacter('\n');
}
void main(){
    code(10);
}
```

2. Schreiben Sie folgende Ninja-code in Ninja-Assembler und zeichen Sie bitte den Stack.

```
writeInteger(func(10, 20, 30);
writeCharacter('\n');

Integer func(Integer a, Integer b, Integer c){
    local Integer resultat;
    resultat = a + b + c;
    return resultat;
}
```

Aufgabe 5

1. Aufgabe: bild1.
2. Aufgabe: bild2.
3. Welche Ausgaben erzeugt das u.a. Programm?

```
void main(int argc, char *argv[]){
    int a[10]={0,1,2,3,4,5,6,7,8,9};
    int *p=&a[4];
    printf("%d\n", a[4]);
    printf("%d\n", *p);
    printf("%d\n", *(p+4)-8);
    printf("%d\n", *(++p));
    printf("%d\n", *(p--));
    printf("%d\n", *(((char *) p)+4));
}
```

4. Aufgabe: bild4.
5. Aufgabe: bild5.

Aufgabe 6: Schreiben Sie bitte die push und pop-methode und prüfen Sie auch die stack-overflow und stack-underflow

```
int stack[1000];
int sp = 0;

void push(int a){
    ...
}
int pop(){
    ...
}
```

Aufgabe 7: Richtig oder Falsch

- Aufgabe: bild6.
- Aufgabe: bild7.

Aufgabe 8: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void funk1(){
    local Integer a;
    a = 5;
    writeInteger(a);
}
```

Aufgabe 9: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void funk2(){
    local Integer a;
    local Integer b;
    a = 10;
    b = 20;
    writeInteger(a+b);
}
```

Aufgabe 10: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
Integer funk3(){
    local Integer a;
    a = 5;
    return a;
}
```

Aufgabe 11: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
Integer funk4(){
    local Integer a;
    local Integer b;
    a = 18;
    b = 20;
    return a+b;
}
```

Aufgabe 12: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void funk5(Integer a){
    writeInteger(a);
}
```

Aufgabe 13: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void funk6(Integer a, Integer b, Integer c, Integer d){
    local Integer e;
    local Integer f;
    e = a + b;
    f = c - d;
    writeInteger(e*f);
}
```

Aufgabe 14: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
Integer funk7(Integer a, Integer b, Integer c, Integer d){
    local Integer e;
    local Integer f;
    e = a + b;
    f = c - d;
    return e*f;
}
```

Aufgabe 15: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void main(){
    writeInteger( funk7(4,3,2,1));

}
```

Aufgabe 16: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code11(){
    local Integer a;
    a = 12;
    writeInteger(a);
    writeCharacter('\n');

}
```

Aufgabe 17: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code12(Integer a){
    if(a>0 && a<=5){
        writeInteger(1);
        writeCharacter('\n');
    }else{
        writeInteger(0);
        writeCharacter('\n');
    }

}
```

Aufgabe 18: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
Integer code13(){
    local Integer a;
    a = 12;
    if((a % 2)==0){
        return 1;
    }else{
        return 1;
    }

}
```

Aufgabe 19: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
Integer code14(Integer a, Integer b){
    if(a>b){
        return a;
    }else{
        return b;
    }

}
```

Aufgabe 20: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
global Integer a;
global Integer b;

void code1(){
    local Integer a;
    a = readInteger();
    if(a<0){
        writeInteger(0);
    }else{
        writeInteger(1);
    }
    writeCharacter('\n');

}
```

Aufgabe 21: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
global Integer a;
global Integer b;

void code2(){
    a = readInteger();
    if((a%2)==0){
        writeInteger(1);
    }else{
        writeInteger(0);
    }
    writeCharacter('\n');
}
```

Aufgabe 22: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code3(){
    local Integer a;
    a = readInteger();
    if((a%2)==0){
        writeInteger(1);
    }else{
        writeInteger(0);
    }
    writeCharacter('\n');
}
```

Aufgabe 23: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code4(){
    a = readInteger();
    b = readInteger();

    if(a>b){
        writeInteger(a);
    }else{
        writeInteger(b);
    }
    writeCharacter('\n');
}
```

Aufgabe 24: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code5(){
    local Integer a;
    local Integer b;
    a = readInteger();
    b = readInteger();

    if(a<b){
        writeInteger(a);
    }else{
        writeInteger(b);
    }
    writeCharacter('\n');
}
```

Aufgabe 25: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code6(){
    a = readInteger();
    while(a>0){
        writeCharacter('a');
        writeCharacter(':');
        writeInteger(a);
        writeCharacter('\n');
        a = a - 1;
    }
    writeCharacter('\n');

}
```

Aufgabe 26: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code7(){
    local Integer a;
    a = readInteger();
    while(a>=0){
        writeCharacter('a');
        writeCharacter(':');
        writeInteger(a);
        writeCharacter('\n');
        a = a - 1;
    }
    writeCharacter('\n');

}
```

Aufgabe 27: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code8(){
    local Integer a;
    local Integer resultat;
    a = readInteger();
    resultat = 1;
    writeInteger(a);
    resultat = 1;
    writeInteger(a);
    writeCharacter('!');
    writeCharacter('=');

    while(a>0){
        resultat = resultat * a;
        a = a - 1;

    }
    writeInteger(resultat);
    writeCharacter('\n');

}
```

Aufgabe 28: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code9(){
    local Integer n;
    local Integer i;
    n = readInteger();
    i = n - 1;
    if (n==0 ){
        writeInteger(0);
    }
    else if (n == 1){
        writeInteger(0);
    }
    else{
        while(i>=2){
            if ( n%i == 0){
                writeInteger(0);
                exit(0);

            }
             i = i - 1;

        }
        writeInteger(1);
        writeCharacter('\n');

    }
}
```

Aufgabe 29: Schreiben Sie folgende Ninja-code in Ninja-Assembler

```
void code10(){
    local Integer n;
    local Integer preprenbr;
    local Integer prenbr,
    local Integer currentnbr,

    n = readInteger();
    preprenbr = 1;
    prenbr = 1;
    currentnbr = 1;

    while(n>=3){
        preprenbr = prenbr;
        prenbr = currentnbr;
        currentnbr = preprenbr + prenbr;
        n = n - 1;
    }
    writeInteger(currentnbr);
    writeCharacter('\n');


}
```