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

Probeklausur KSP

Aufgabe 1

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.)

```c
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:

```c
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

```c
/** 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 2

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 3

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 4: 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 5: Richtig oder Falsch

- Aufgabe: bild6.
- Aufgabe: bild7.