

1. Stack Memory

Fill in the memory contents at the two lines labeled *STACK DUMP*. Assume a simple stack memory model where variables on stack frames for new functions immediately follow the last allocated variable from the calling function, and that no extra space is needed for meta data. Draw a heavy line between stack frames and label with the called function. Indicate memory that has unknown contents or should be inaccessible.

Stack Dump 1		
Address	Name	Contents
10000		
10004		
10008		
10012		
10016		
10020		
10024		
10028		
10032		

main()

Stack Dump 2		
Address	Name	Contents
10000		
10004		
10008		
10012		
10016		
10020		
10024		
10028		
10032		

main()

Code

```
int main(void) {
    int a, b, c;
    a = 123;
    b = 456;
    c = func_2(a);
    func_3();
    // STACK DUMP 2
}

int func_2(int x) {
    int y = 15;
    x = 100;
    func_3();
    // STACK DUMP 1
    return x + y;
}

void func_3(void) {
    int array[3];
    array[0] = -5;
}
```

2. Pointer Syntax

```
int i;  
int j;  
int *p;
```

Given the variable declarations above, what syntax would you use to access the following pieces of information in C code?

The value of i:

The address that p points to:

The address of j:

The address of p:

The value that p points to:

3. Pointer Memory

If the code below is run, what will be the memory contents at STACK DUMP 3?

```
int main()  
{  
    int a = 10;  
    int *ptr;  
    ptr = &a;  
    *ptr = 20;  
    // STACK DUMP 3  
    ...  
}
```

Stack Dump 3		
Address	Name	Contents
10000		
10004		
10008		