1. Given the array declaration

```c
char a[4][5];
```

Mark with an A the memory location where we would find

```
123456789012345678901234567890
(a[2][4]      a[2,4])
```

2. Show the memory layout of the following struct/record definition taking into consideration the SPARC data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate struct/record member/field name. For example, if member/field name p takes 4 bytes, you will have 4 p's in the appropriate memory locations. If the member/field is an array, use the name followed by the index number. For example, some number of p0's, p1's, p2's, etc. Place an X in any bytes of padding.

```c
struct foo { double a; fubar: char b; int c[3]; short d; long e; }
```

```c
struct foo fubar;
```

What is the `offsetof(struct foo, d)`?

What is the `sizeof(struct foo)`?
3. Fill in the blanks

A ____________ subroutine always has a stack frame associated with it when it is called.

A ____________ subroutine supports recursion.

A ____________ subroutine does not (cannot) call other subroutines.

A ____________ subroutine usually results in a larger code size compared to equivalent code implemented as other types of subroutines.

A ____________ subroutine shares the same register window set as its caller.

In the SPARC architecture, the ________________ register points into the current/active stack frame such that ________________ variables are accessed with a ________________ offset and ________________ are accessed with a ________________ offset.

4. Indicate whether the following expressions are

A. legal (no compiler error) or
B. illegal (compiler error).

```c
int a[10];
int *iPtr;

a = iPtr;    ______ iPtr = a;    ______
*iptr = a[3]; ______ a[3] = *iPtr; ______
3[a] = 5; ______ *a = iPtr[2]; ______
```

What question would you most like to see on the Midterm?