

Login name _____

Quiz 3

Name _____

CSE 131B

Signature _____

Spring 2007

Student ID _____

1. What gets printed at each printf() statement given the following C program

```
#include <stdio.h>

int
main()
{
  char s[] = "absolute";
  char *p = s;

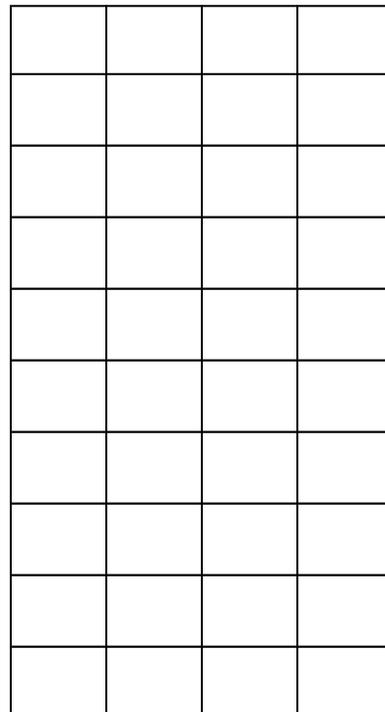
  printf( "%c\n", *p++ ); _____
  --*(p+4);
  printf( "%c\n", *++p ); _____
  p = p+1;
  *p = *(p-3) + 4;
  printf( "%c\n", p[0] ); _____
  *(p+1) = p[1] + 2;
  printf( "%c\n", *++p ); _____
  p++;
  printf( "%c\n", *p++ ); _____
  p[0] = *(p+1);
  printf( "%s\n", s ); _____
  return 0;
}
```

2. Show the memory layout of the following C 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 *p0s*, *p1s*, *p2s*, etc. Place an *X* in any bytes of padding. Structs and unions are padded so the total size is evenly divisible by the most strict alignment requirement of its members.

```
struct foo {
  char   a;
  short  b[5];
  double c;
  int    d;
};

struct foo fubar;
```

fubar:



low memory

high memory

What is the sizeof(struct foo)? _____

What is the offsetof(struct foo, b[4])? _____

If struct foo had been defined as union foo instead, what would be the sizeof(union foo)? _____

3. Give an example of a non-converting type cast (underlying bit pattern does not change).

Give an example of a converting type cast (underlying bit pattern does change).

4. For the following Oberon statements, indicate the correct error message using the list of given error messages below (if there is no error, select option A):

Possible Error Messages:

A - No error

B - BOOLEAN required for conditional test

C - Argument not assignable to value parameter

D - Argument not equivalent to REF parameter

E - Non-addressable argument passed to REF parameter

F - Incompatible type to binary operator

G - Incompatible type to unary operator

H - Left hand side of assignment statement is not assignable (not a modifiable L-value)

I - Array index out of bounds

```
CONST t = 3;
TYPE foo = INTEGER;
TYPE bar = FLOAT;
TYPE baz = BOOLEAN;
VAR w : ARRAY 5 OF foo;
VAR x : POINTER TO foo;
VAR y : bar;
VAR z : baz;
FUNCTION p(a : INTEGER; REF b : FLOAT) : foo;
    RETURN 0;
END p;
```

```
BEGIN
    y := p(w[4], y); _____
    x^ := p( p( t, y ), 4.20 ); _____
    y^ := w[t]; _____
    p(x^, x^); _____
END.
```

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