CMSC 330 Exam 2: Nov. 22, 1999

Show all work necessary to justify your answers!

1. (20 points)
   What will the following program, in C++ syntax, output if parameters are passed with:

   - call-by-value
   - call-by-value-result
   - call-by-reference
   - call-by-name

```
int i, j;

int main(int argc, char* argv[])
{
  i = 1;
  j = 2;
  f(i, j);
  cout << i << " " << j << '\n';
}

void f(int j, int k)
{
  j = j - 9;
  k = k + 8;
  i = i - j;
}
```

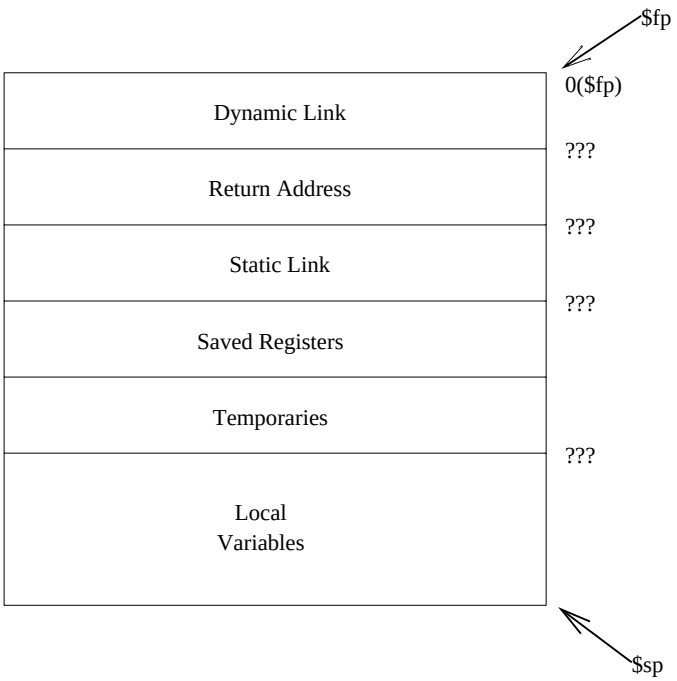
   value: 9 2
   value-result: -8 10
   reference: 0 10
   name: 0 10

2. (50 points total)
This question requires you to write pseudo-code for several MIPS assembly language instructions that must be executed to run the following program. Your pseudo-code should be as detailed as the assembly instructions, but we are not concerned about syntax. For example, the instruction "sw $fp, 0($sp)" can be written as "store the frame pointer at 0 past the stack pointer."

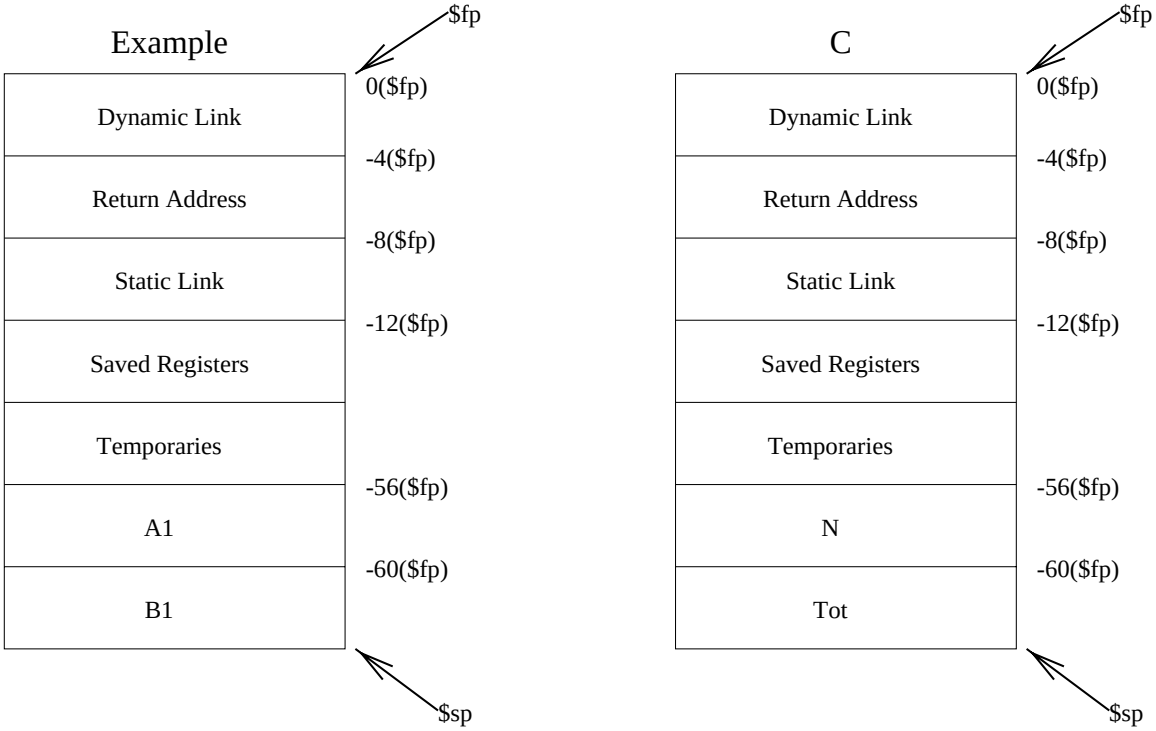
```
program Example ;
  var A1,B1: integer;

  procedure C(var N,Tot: integer);

    procedure Show(Value:integer);
      begin
        writeln(Value);
        writeln(Tot);
        writeln(B1)
      end;

    begin
      if (N >= 1) then
        begin
          Tot := Tot * N;
          N := N - 1;
          C(N,Tot)
        end
      else
        Show(Tot)
    end;

begin
  A1 := 4;
  B1 := 1;
  C(A1,B1)
end.
```

You should assume the register conventions discussed in class, and parameter passing is done as was shown in class for Pascal (i.e. default is pass-by-value, unless explicitly specified pass-by-reference in a procedure header with `var`).

The template for a single stack frame (activation record instance) is shown below.



(a) (10 points)
Show a complete layout for the stack frames (activation record instances) for `Example()` and `C()`, based on the template. Fill in all the missing relative addresses (marked by ??? in the template), and make sure to show all local variables, and their locations in the stack frame relative to the frame pointer. Remember that, by convention, the saved registers and temporaries use 44 bytes.



(b) (30 points)
Provide the pseudo-code that is necessary to access the values of the arguments to writeln in the three writeln statement in procedure Show:

- writeln(Value);
- writeln(Tot);
- writeln(B1)

Use the stack frame layout from the first part of this question to obtain the addresses of local variables.

Value is local and passed by value:

```
la $t0, -56($fp)    # only local variable in Show
lw $t0, ($t0)
```

Tot is non-local (from C) and pass-by-reference to C:

```
lw $t0, -8($fp)     # static link to ARI of C
lw $t0, -60($t0)    # Tot
lw $t0, ($t0)       # follow pointer to B1
```

B1 is non-local (from Example), so need to follow 2 static links

```
lw $t0, -8($fp)     # static link to ARI for C
lw $t0, -8($t0)     # static link to ARI for Example
lw $t0, -60($t0)    # B1
```

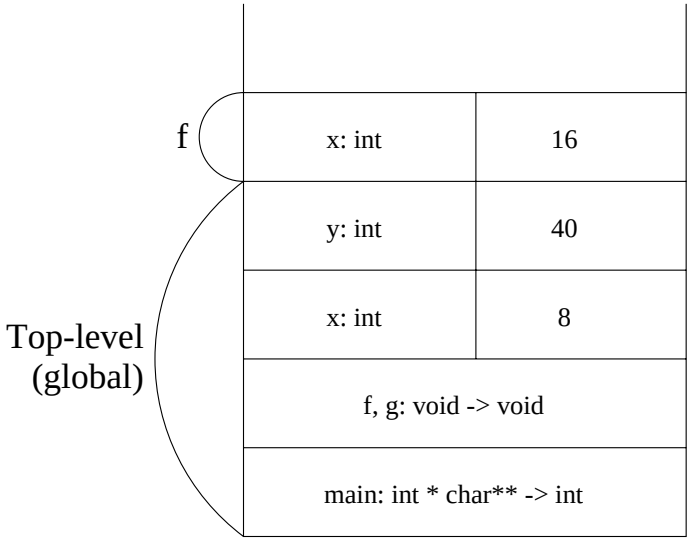
(c) (10 points)
What does the program print?

24
24
24

3. (30 points total)

(a) (10 points)
If parameters are passed by value, and using static scoping rules, what does the environment (name/value pairs, including functions) look like just before function g returns, for the following program, which uses C++ syntax?

```
int x, y;

int main(int argc, char* argv[])
{
  x = 2;
  y = 10;
  f();
  cout << x << " " << y << '\n';
}

void g()
{
  x = y / 5;
}

void f()
{
  int x;
  x = y + 6;
  y = 4 * y;
  g();
  cout << x << '\n';
}
```



(b) (20 points)
What does the program output with:

- static scoping,
- dynamic scoping

for non-local variables.

static :
16
8 40
dynamic:
8
2 40