1. For each of the three parameter transition techniques call–by–value, call–by–reference, and call–by–name, give the output of the following program, written using C syntax.

```
#include <stdio.h>

int a, b, i;
int c[4]= {100, 150, 175, 190};

void f(int d, int e, int f) {
  d= 5;
  b= a + 3;
  e= 3;
  i= 0;
  f= 20;
  c[i]= 200;
}

int main() {
  a= 10;
  b= 5;
  i= 1;
  f(a, i, c[i]);
  printf("%d %d %d %d %d %d %d\n", a, b, i, c[0], c[1], c[2], c[3]);
  return 0;
}
```

2. Consider the following code written in OCaml syntax:

```
let value = ref 1;;

let f n = value := !value +  n ; !value;;

let g v w =
  let x = (f 2) in
    let y = v + w in
      let z = v * w in
        x + z - y;;

g (f 1) (if !value > 4 then 6 else 5)
```

   (a) What output would the program produce if parameters were passed by value (as they actually are in OCaml)?

   (b) What output would the program produce if parameters were passed by need, as they are in Haskell? Recall that in call–by–need an actual parameter is evaluated only once, at the point where it is first used in the called procedure.

   (c) What output would the program produce if parameters were passed by name?

3. What results would the following program, written in C syntax, produce if parameters were passed by need? If C adopted call–by–need as its default parameter passing mechanism, what would the program's results be?

```
#include <stdio.h>

int func(int a, int b) {
  if (b == 0)
    return 0;
  else return func(a, b);
}

int main() {
  printf("%d\n", func(func(1, 1), func(0, 0)));
  return 0;
}
```

4. The following procedure will exchange the values of two integer variables if they are passed by reference. If the parameter transmission mechanism were changed to call–by–name, is there a set of input variables whose values cannot be exchanged?

```
void swap(x, y: integer);
  int temp;
  temp= x;
  x= y;
  y= temp
}
```

5. Give the least restrictive limits that can be placed on actual parameters so that call–by–name and call–by–reference always give the same results in all cases.