1. Parameter Passing (5 + 5 + 5 + 5 = 20 points)
Consider the following Ada program:

```ada
with Ada.Text_IO;
procedure Main is
  A : Character;
  procedure P(B : in out Character) is begin B := 'x'; A := 'y'; end P;
begin
  A := 'z';
  P(A);
  Ada.Text_IO.Put_Line(Character'Image(A));
end Main;
```

1a. (5 points) What does the program print if “in out” means call-by-value?
1b. (5 points) What does the program print if “in out” means call-by-value-result?
1c. (5 points) What does the program print if “in out” means call-by-reference?
1d. (5 points) Which parameter passing mode does Ada use for “in out” Integer parameters?

2. Exceptions (10 + 10 = 20 points)
2a. (10 points) What is the difference between checked and unchecked exceptions in Java? What is the advantage of using a checked exception? What is the advantage of using an unchecked exception?
2b. (10 points) What is a `finally` clause in Java? What is it good for?

3. Type Inference (5 + 5 + 10 = 20 points)
Consider the following OCaml function:

```ocaml
let rec g = function [] -> [] | h::t -> (h + 2)::(g t);
```

3a. (5 points) Describe in words what g does.
3b. (5 points) What is the type of g?
3c. (10 points) Show the type inference steps for g.
4. Virtual Methods (5 + 5 + 5 + 5 = 20 points)

Consider the following Java program:

```java
class C0 {
    int a() { return 0; } int b() { return 1; }
}
class C1 extends C0 {
    int cee() { return 12; } int b() { return 11; }
}
class C2 extends C1 {
    int a() { return 20; } int b() { return 21; }
}
class Main {
    public static void main(String[] args) {
        C0[] x = new C0[3];
        x[0] = new C0(); x[1] = new C1(); x[2] = new C2();
        for (int i=0; i<x.length; i++)
        { System.out.println(x[i].a()); System.out.println(x[i].b()); }
    }
}
```

4a. (5 points) What is the vtable of class C0?
4b. (5 points) What is the vtable of class C1?
4c. (5 points) What is the vtable of class C2?
4d. (5 points) What does the program print?

5. Concurrency (10 + 5 + 5 = 20 points)

Consider the following (buggy) bounded buffer put method:

```java
public void put(String s) {
    while (count >= buf.length) { /* do nothing */ }
    count++;
    in++;
    buf[in % buf.length] = s;
}
```

Assume that `buf.length` is 3, and that each line of code in the `put` method is atomic.

5a. (10 points) Show an interleaving of two calls to `put` with an undesirable race condition. In other words, there are two producer threads, both of which are executing `put` at the same time. Assume that they pass the strings "s1" and "s2" as parameters to `put`. Your answer should be a table that starts like this:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>count</th>
<th>in</th>
<th>thread: code</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>0</td>
<td>-1</td>
<td>i: while(count&gt;=buf.length) {}</td>
</tr>
</tbody>
</table>

5b. (5 points) Fix the code to prevent the race condition.
5c. (5 points) Fix the code from (b) to prevent deadlock.