Semantic difference - any reasonable “semantic” difference works
For example,

a. reference is just an alias for an “already existing” object, while pointer points to a certain object.
b. A pointer can be re-assigned, A reference cannot be re-bound, and must be bound at initialization.
c. A pointer variable has its own identity: a distinct, visible memory address that can be taken with the unary & operator and a certain amount of space that can be measured with the sizeof operator. Using those operators on a reference returns a value corresponding to whatever the reference is bound to; the reference’s own address and size are invisible.
d. You can have arbitrarily nested pointers to pointers offering extra levels of indirection. References only offer one level of indirection because references to references collapse.
e. A pointer can be assigned nullptr, whereas a reference must be bound to an existing object. If you try hard enough, you can bind a reference to nullptr, but this is undefined and will not behave consistently.
f. Const references and rvalue references can be bound to temporaries (see temporary materialization). Pointers cannot (not without some indirection):

```
const int &x = int(12); // legal C++
int *y = &int(12); // illegal to take the address of a temporary.
```

Any of these could also work for when one is preferable over another if the correct explanation is also offered. No marks without explanation. For example - “Pointers are used in linked lists while references are used in functions” is NOT a valid answer - if explanation is given then it is fine.