Princeton University
COS 217: Introduction to Programming Systems
C Unions

Problem: We need to define an array of 10 elements, some of which are of type int and some of which are of type double. What should the element type be?

Solution 1: The generic pointer (void*)

```
void *a[10];
int *pi;
double *pd;
...
pi = (int*)malloc(sizeof(int));
*pi = 5;
a[0] = pi;
pd = (double*)malloc(sizeof(double));
*pd = 5.5;
a[1] = pd;
...
printf("%d", *(int*)a[0]);
printf("%f", *(double*)a[1]);
...
free(a[0]);
free(a[1]);
...
```

Note: Awkward

Solution 2: A structure

```
struct MyStruct
{
   int i;
   double d;
};
...
struct MyStruct a[10];
...
a[0].i = 5;
a[1].d = 5.5;
...
printf("%d", a[0].i);
printf("%d", a[1].d);
...
```

Note: Wastes space
Note: For each element, how do we know whether i or d is significant?

Solution 3: A union

```
union MyUnion
{
   int i;
   double d;
};
...
union MyUnion a[10];
...
a[0].i = 5;
a[1].d = 5.5;
...
printf("%d", a[0].i);
printf("%d", a[1].d);
...
```

Note: For each element, how do we know whether i or d is significant? (Disastrous to choose the wrong field.)

Solution 4: A structure containing a "kind" field and a union

```
enum Kind = {INTEGER, DOUBLE};
struct MyStruct
{
   enum Kind k;
   union
   {
      int i;
      double d;
   } u;
};
...
struct MyStruct a[10];
...
a[0].k = INTEGER;
a[0].u.i = 5;
a[1].k = DOUBLE;
a[1].u.d = 5.5;
...
if (a[0].k == INTEGER)
   printf("%d", a[0].u.i);
else
   printf("%f", a[0].u.d);
if (a[1].k == INTEGER)
   printf("%d", a[1].u.i);
else
   printf("%f", a[1].u.d);
```