Value	Representation (bits)	Representation		Useful C type
	31-bit, 2's complement number	hex de	decimal	
9	0000 0000 0000 0000 0000 0001 0011	0x000000 13	19	int
-2	1111 1111 1111 1111 1111 1111 1111 1101	0xfffffff D	-3	int
true	0000 0000 0000 0000 0000 0000 0000 0110	0x00000006	6	int
false	0000 0000 0000 0000 0000 0000 0000 0010	0x00000002	2	int
(pair 1 2)	XXXX XXXX XXXX XXXX XXXX XXXX XXXX XX 00	0xXXXXXXXX	big num	
	tag hits (V1-num	10-hool 00-r	nair)	

tag bits (X1=num, 10=bool, 00=pair)

```
extern int our_code_starts_here()
                                                                     union snake_val {
 asm("our_code_starts_here");
                                                                       int as_int;
                                                                       union snake_val* as_ptr;
void print_val(int val) {
  if(val & 1) { printf("%d", (val - 1) / 2); }
                                                                     extern union snake_val our_code_starts_here()
  else if (val == 6) { printf("true"); }
                                                                       asm("our_code_starts_here");
  else if (val == 2) { printf("false"); }
  else {
                                                                     void print_val(union snake_val val) {
                                                                       if(val.as_int & 1) {
                                                                         printf("%d", (val.as_int - 1) / 2);
                                                                       else if (val.as_int == 6) { printf("true"); }
                                                                       else if (val.as_int == 2) { printf("false"); }
                                                                       else { // It's a pair!
  }
int main(int argc, char** argv) {
  int input = 0;
                                                                       }
  int* MEMORY = calloc(10000, sizeof(int));
                                                                     int main(int argc, char** argv) {
                                                                       int input = 0;
  if(argc > 1) { input = atoi(argv[1]); }
                                                                       int* MEMORY = calloc(10000, sizeof(int));
  int result = our_code_starts_here(input, MEMORY);
                                                                       if(argc > 1) { input = atoi(argv[1]); }
  print_val(result);
                                                                       union snake_val result;
  printf("\n");
                                                                       result = our_code_starts_here(input, MEMORY);
  fflush(stdout);
                                                                       print_val(result);
  return 0;
                                                                       printf("\n");
                                                                       fflush(stdout);
                                                                       return 0;
                                                                     }
```

```
asm("our_code_starts_here");
                                                                      int as_int;
                                                                      union snake_val* as_ptr;
void print_val(int val) {
 if(val & 1) { printf("%d", (val - 1) / 2); }
                                                                    extern union snake_val our_code_starts_here()
 else if (val == 6) { printf("true"); }
                                                                      asm("our_code_starts_here");
 else if (val == 2) { printf("false"); }
 else {
                                                                    void print_val(union snake_val val) {
                                                                      if(val.as_int & 1) {
                                                                        printf("%d", (val.as_int - 1) / 2);
                                                                      else if (val.as_int == 6) { printf("true"); }
                                                                      else if (val.as_int == 2) { printf("false"); }
                                                                      else { // It's a pair!
int main(int argc, char** argv) {
 int input = 0;
                                                                      }
                                                                    }
 int* MEMORY = calloc(10000, sizeof(int));
                                                                    int main(int argc, char** argv) {
                                                                      int input = 0;
 if(argc > 1) { input = atoi(argv[1]); }
                                                                      int* MEMORY = calloc(10000, sizeof(int));
 int result = our_code_starts_here(input, MEMORY);
                                                                      if(argc > 1) { input = atoi(argv[1]); }
 print_val(result);
                                                                      union snake_val result;
 printf("\n");
                                                                      result = our_code_starts_here(input, MEMORY);
 fflush(stdout);
                                                                      print_val(result);
 return 0;
                                                                      printf("\n");
                                                                      fflush(stdout);
                                                                      return 0;
                                                                    }
  Value
                                     Representation (bits)
                                                                      Representation
                                                                                                    Useful C type
                                                                      hex
                                                                                decimal
                           31-bit, 2's complement number
        9
                         0000 0000 0000 0000 0000 0000 0001 0011
                                                                                                         int
                                                                     0x00000013
                                                                                   19
       -2
                         1111 1111 1111 1111 1111 1111 1111 1101
                                                                                                         int
                                                                    0xfffffffD
                                                                                   -3
                         0000 0000 0000 0000 0000 0000 0000 0110
                                                                                                         int
     true
                                                                     0x00000006
    false
                         0000 0000 0000 0000 0000 0000 0000 0010
                                                                                                         int
                                                                     0x00000002
```

union snake_val {

extern int our_code_starts_here()

(pair 1 2)

tag bits (X1=num, 10=bool, 00=pair)

0xXXXXXXXX

big num

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXX00