Listing 1: `ar_stack.c`

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef int item_t;
5
6  typedef struct {item_t *base; item_t *top; int size;} stack_t;
7
8  stack_t *create_stack(int size)
9  {    stack_t *st;
10      st = (stack_t *) malloc( sizeof(stack_t) );
11      st->base = (item_t *) malloc( size * sizeof(item_t) );
12      st->size = size;
13      st->top = st->base;
14      return( st );
15 }
16
17 int stack_empty(stack_t *st)
18 {    return( st->base == st->top );
19 }
20
21 int push( item_t x, stack_t *st)
22 {    if ( st->top < st->base + st->size )
23      {    *(st->top) = x; st->top += 1;  return( 0 );
24      }
25      else
26         return( -1 );
27 }
28
29 item_t pop(stack_t *st)
30 {    st->top -= 1;
31      return( *(st->top) );
32 }
33
34 item_t top_element(stack_t *st)
35 {    return( *(st->top -1) );
36 }
37
38 void remove_stack(stack_t *st)
39 {    free( st->base );
40      free( st );
41 }
42
43
44
```

```
45 int main()
46 {  stack_t *st;
47    char nextop;
48    st = create_stack(50);
49    printf("Made Array-Based Stack of size 50\n");
50    while( (nextop = getchar())!= 'q' )
51    { if( nextop == 'i' )
52      { int insitem;
53        scanf(" %d", &insitem);
54        push( insitem, st );
55        printf(" pushed %d. The current top item is %d\n", insitem,
56            top_element(st) );
57      }
58      if( nextop == 'd' )
59      { int de_item;
60        getchar();
61        de_item = pop(st);
62        printf("  popped item %d", de_item);
63        if( stack_empty(st) )
64          printf(" the stack is now empty\n");
65        else
66          printf(" the top element is now %d\n",  top_element(st) );
67
68      }
69      if( nextop == '?' )
70      { getchar();
71        if( stack_empty(st) )
72          printf("the stack is empty\n");
73        else
74          printf("the top element is %d\n", top_element(st) );
75      }
76
77    }
78    remove_stack(st);
79    printf(" removed stack\n");
80    return(0);
81 }
```