```
static void terminate(const char *message)
{
  printf("%s\n", message);
  exit(EXIT_FAILURE);
}

void make_empty(void)
{
  while (!is_empty())
    pop();
}

bool is_empty(void)
{
  return top == NULL;
}

bool is_full(void)
{
  return false;
}

void push(int i)
{
  struct node *new_node = malloc(sizeof(struct node));
  if (new_node == NULL)
    terminate("Error in push: stack is full.");

  new_node->data = i;
  new_node->next = top;
  top = new_node;
}

int pop(void)
{
  struct node *old_top;
  int i;

  if (is_empty())
    terminate("Error in pop: stack is empty.");

  old_top = top;
  i = top->data;
  top = top->next;
  free(old_top);
  return i;
}
```

Note that the `is_full` function returns `false` every time it's called. A linked list has no limit on its size, so the stack will never be full. It's possible (but not likely) that the program might run out of memory, which will cause the `push` function to fail, but there's no easy way to test for that condition in advance.

Our stack example shows clearly the advantage of information hiding: it