I've also added a new member, `size`, that stores the stack's maximum size (the length of the array that `contents` points to). We'll use this member to check for the "stack full" condition.

The `create` function will now have a parameter that specifies the desired maximum stack size:

```
Stack create(int size);
```

When `create` is called, it will create a `stack_type` structure plus an array of length `size`. The `contents` member of the structure will point to this array.

The `stackADT.h` file will be the same as before, except that we'll need to add a `size` parameter to the `create` function. (Let's name the new version `stackADT2.h`.) The `stackADT.c` file will need more extensive modification, however. The new version appears below, with changes shown in **bold.**

stackADT2.c

```
#include <stdio.h>
#include <stdlib.h>
#include "stackADT2.h"

struct stack_type {
  Item *contents;
  int top;
  int size;
};

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

Stack create(int size)
{
  Stack s = malloc(sizeof(struct stack_type));
  if (s == NULL)
    terminate("Error in create: stack could not be created.");
  s->contents = malloc(size * sizeof(Item));
  if (s->contents == NULL) {
    free(s);
    terminate("Error in create: stack could not be created.");
  }
  s->top = 0;
  s->size = size;
  return s;
}

void destroy(Stack s)
{
  free(s->contents);
  free(s);
}
```