Recitation 5 Answer

From Xiao Liu

Exercise 1.1

```c
#include <stdio.h>

int main(void) {
  n += 5;
  printf("%d\n", n);

  return 0;
}
```

- It does **NOT** compile since `n` is not declared / initialized in the scope of `main()`.

Exercise 1.2

```c
#include <stdio.h>

int n = 0;

int main(void) {
  n += 5;
  printf("%d\n", n);

  return 0;
}
```

- It compiles since `n` has file scope.
- The output is `5`.

Exercise 1.3

```c
#include <stdio.h>

int n = 0;

int main(void) {
  int n;
  n += 5;

  printf("%d\n", n);

  return 0;
}
```

- It compiles since there is an `n` is with file scope and an `n` with local scope.
- Print the local scope `n` out is an undefined behavior since it's haven't been initialized before used.

Exercise 1.4

```c
#include <stdio.h>

int n = 0;

int main(void){
  int n = 1;

  n += 5;
  printf("%d\n", n);

  return 0;
}
```

- It compiles since there is an `n` is with file scope and an `n` with local scope.
- The output is `6`. The inner scope hides the outer scope.

Exercise 1.5

```c
#include <stdio.h>

int n = 0;

int main(void) {
  extern int n;

  n += 5;
  printf("%d\n", n);

  return 0;
}
```

- It compiles and outputs `5`.

Exercise 1.6

```c
#include <stdio.h>

int total(int x) {
  static int total = 0;

  total += x;
  return total;
}

int main(void) {
  total(5);
  total(10);
  printf("%d\n", total(15));
  return 0;
}
```

- It compiles.
- The output is `30`.

Exercise 1.7

`library.h`:

```c
#ifndef LIBRARY_H
#define LIBRARY_H

int net_price(int price, int discount);
static int validate(int n);

#endif
```

`driver.c`:

```c
#include "library.h"
int main(void) {
  int i, j;
  i = validate(-5);
  j = net_price(6, 2);
}
```

`library.c`:

```c
#include "library.h"

int net_price(int price, int discount) {
  return validate(price - discount);
}

static int validate(int n) {
  if (n < 0)
    return 0;
  else
    return n;
}
```

- It compiles.
- It does **NOT** link since the reference to `validate()` in `main()` is undefined because `validate()` has internal linkage.

Exercise 2.1

`account.h`:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#define ACCT_OK      0
#define ACCT_INVALID 1
#define ACCT_CLOSED  2

int acct_status;

#endif
```

`account.c`:

```
#include "account.h"

// Use acct_status
```

`driver.c`:

```
#include "account.h"

// Use acct_status
```

- It compiles, because `account.h` is in both translation units.
- It does **NOT** link because `acct_status` has been defined in both translation units.

Exercise 2.2

`account.h`:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#define ACCT_OK       0
#define ACCT_INVALID 1
#define ACCT_CLOSED  2

// Don't define acct_status
// since that caused the problem

#endif
```

`account.c`:

```
#include "account.h"

// Use acct_status
```

`driver.c`:

```
#include "account.h"

// Use acct_status
```

- It does **NOT** compile and you will get a message like 'acct_status' was not declared in this scope.

Exercise 2.3

`account.h`:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#define ACCT_OK      0
#define ACCT_INVALID 1
#define ACCT_CLOSED  2

// Don't define acct_status

#endif
```

`account.c`:

```
#include "account.h"

int acct_status;

// Use acct_status
```

`driver.c`:

```
#include "account.h"

// Use acct_status
```

- `account.c` will compile but `driver.c` won't.

Exercise 2.4

`account.h`:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#define ACCT_OK      0
#define ACCT_INVALID 1
#define ACCT_CLOSED  2

extern int acct_status;

#endif
```

`account.c`:

```
#include "account.h"

// Use acct_status
```

`driver.c`:

```
#include "account.h"

// Use acct_status
```

- They compile, because `acct_status` is declared in `account.h`.
- They do **NOT** link since `acct_status` has been declared but not defined.
  - `undefined reference to 'acct_status'`

Exercise 2.5

`account.h`:

```
#ifndef ACCOUNT_H
#define ACCOUNT_H

#define ACCT_OK      0
#define ACCT_INVALID 1
#define ACCT_CLOSED  2

extern int acct_status;

#endif
```

`account.c`:

```
#include "account.h"

int acct_status = ACCT_OK;

// Use acct_status
```

`driver.c`:

```
#include "account.h"

// Use acct_status
```

- Correct implementation.

Recitation 5 Answer

From Xiao Liu