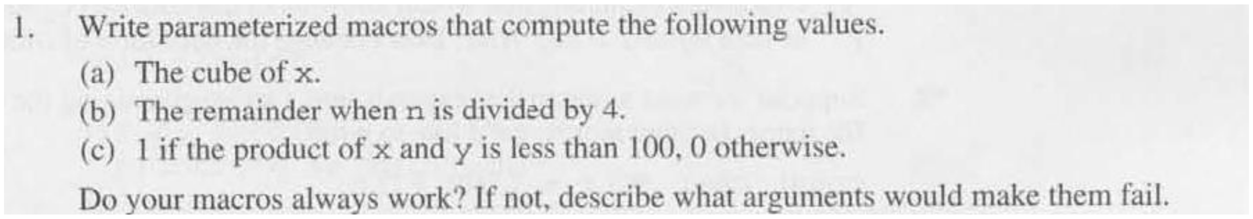
York University
Electrical Engineering and Computer Science

EECS2031: Software Tools
SU2016
Assignment #8

Chapter 14: Exercises

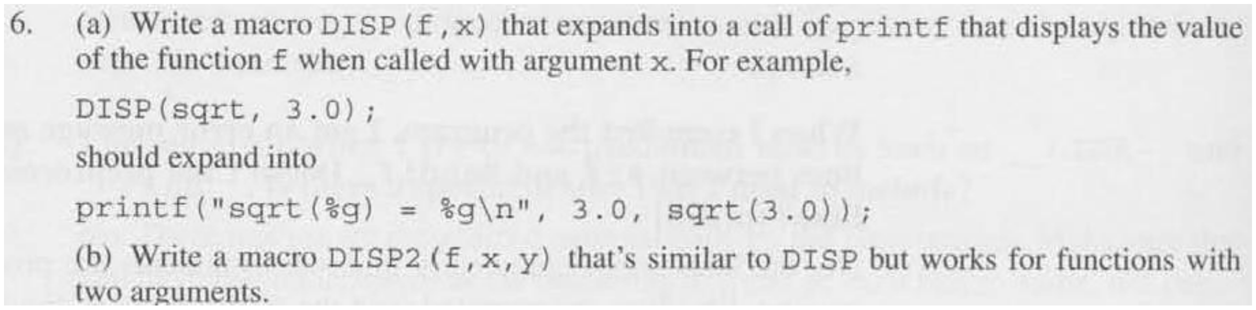

1. Write parameterized macros that compute the following values.
   (a) The cube of `x`.
   (b) The remainder when `n` is divided by 4.
   (c) 1 if the product of `x` and `y` is less than 100, 0 otherwise.

   Do your macros always work? If not, describe what arguments would make them fail.

(a) `#define CUBE(x) ((x) * (x) * (x))`
This macro may not work correctly if `x` has a side effect.

(b) `#define MOD4(n) ((n) % 4)`

(c) `#define LT100(x, y) ((x) * (y) < 100)`
This macro may not work correctly if `x` has a side effect that affects `y`, or vice-versa.


6. (a) Write a macro `DISP(f,x)` that expands into a call of `printf` that displays the value of the function `f` when called with argument `x`. For example,

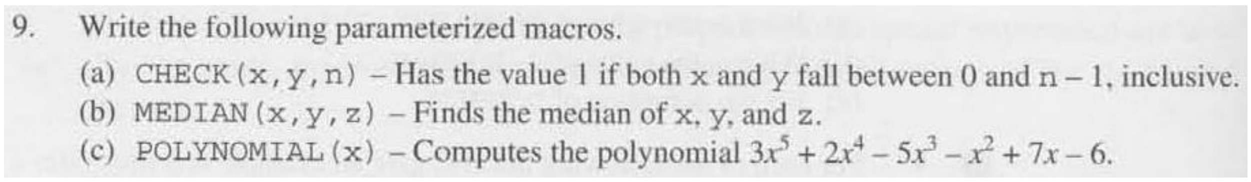
   ```
   DISP(sqrt, 3.0);
   ```

   should expand into

   ```
   printf("sqrt(%g) = %g\n", 3.0, sqrt(3.0));
   ```

   (b) Write a macro `DISP2(f,x,y)` that's similar to `DISP` but works for functions with two arguments.

(a) `#define DISP(f, x) printf(#f "(%g) = %g\n", x, f(x))`
(b) `#define DISP2(f, x, y) printf(#f "(%g, %g) = %g\n", x, y, f(x, y))`


9. Write the following parameterized macros.
   (a) `CHECK(x,y,n)` – Has the value 1 if both `x` and `y` fall between 0 and `n` – 1, inclusive.
   (b) `MEDIAN(x,y,z)` – Finds the median of `x`, `y`, and `z`.
   (c) `POLYNOMIAL(x)` – Computes the polynomial $3x^5 + 2x^4 - 5x^3 - x^2 + 7x - 6$.

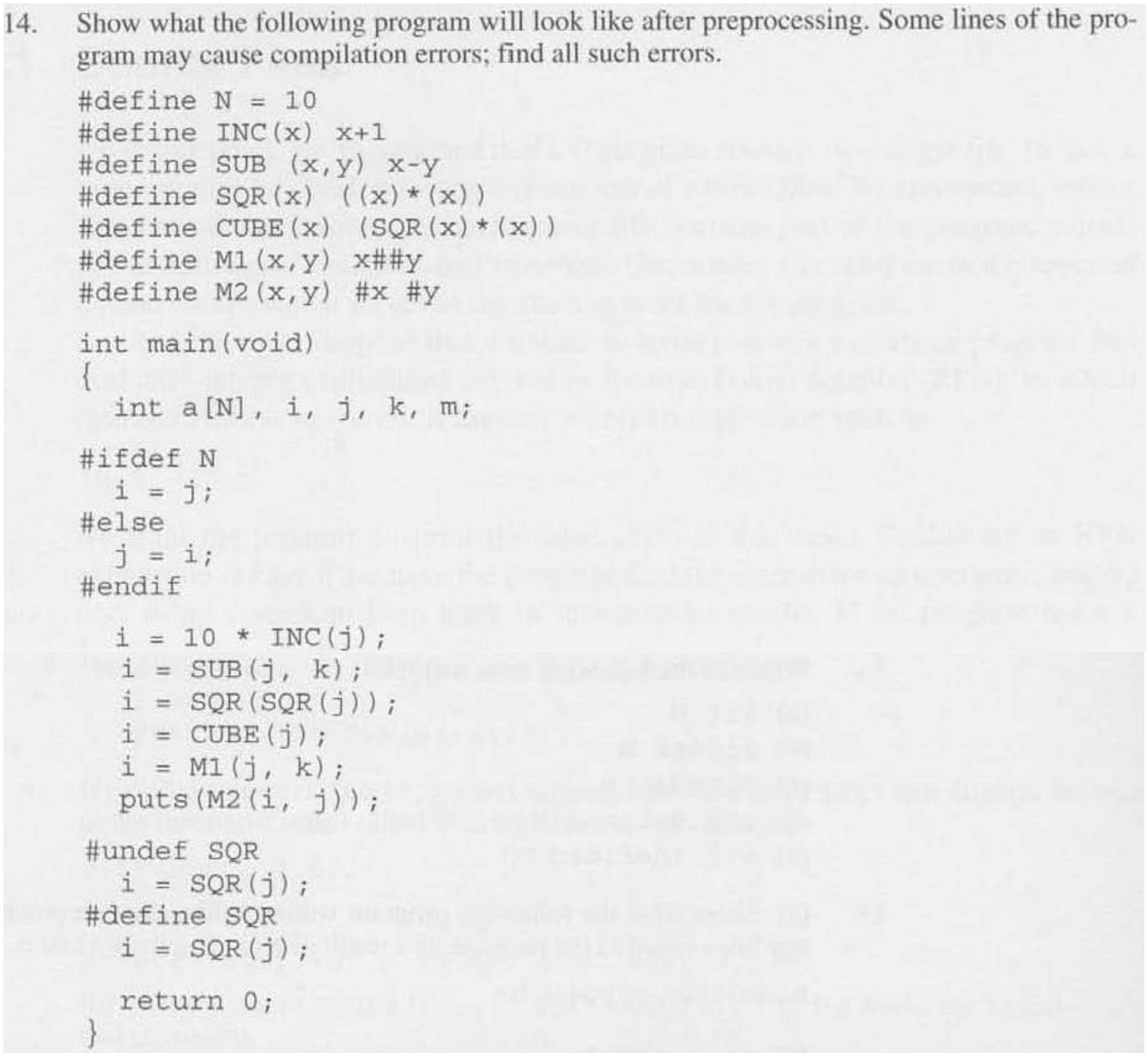
(a) `#define CHECK(x, y, n) (0 <= (x) && (x) <= (n) - 1 && 0 <= (y) && (y) <= (n) - 1)`

(b)

```
#define MAX(x, y) ((x) > (y) ? (x) : (y))
#define MIN(x, y) ((x) < (y) ? (x) : (y))
#define MEDIAN(x, y, z) ((x) > (y) ? ((x) > (z) ? MAX(y, z) : (x)) : ((x) >
(z) ? (x) : MIN(y, z)))
```

(c)

```
#define POLYNOMIAL(x) (((((3.0 * (x) + 2.0) * (x) - 5.0) * (x) - 1.0) *
(x) + 7.0) * (x) - 6.0)
```



14. Show what the following program will look like after preprocessing. Some lines of the program may cause compilation errors; find all such errors.

```
#define N = 10
#define INC(x) x+1
#define SUB (x,y) x-y
#define SQR(x) ((x)*(x))
#define CUBE(x) (SQR(x)*(x))
#define M1(x,y) x##y
#define M2(x,y) #x #y

int main(void)
{
  int a[N], i, j, k, m;

#ifdef N
  i = j;
#else
  j = i;
#endif

  i = 10 * INC(j);
  i = SUB(j, k);
  i = SQR(SQR(j));
  i = CUBE(j);
  i = M1(j, k);
  puts(M2(i, j));

#undef SQR
  i = SQR(j);
#define SQR
  i = SQR(j);

  return 0;
}
```

Blank line
Blank line
Blank line
Blank line
Blank line
Blank line
Blank line

```
int main(void)
{
  int a[= 10], i, j, k, m;

Blank line
  i = j;
Blank line
Blank line
Blank line

  i = 10 * j+1;
  i = (x,y) x-y(j, k);
  i = ((((j)*(j)))*(((j)*(j))));
  i = (((j)*(j))*(j));
  i = jk;
  puts("i" "j");

Blank line
  i = SQR(j);
Blank line
  i = (j);

  return 0;
}
```

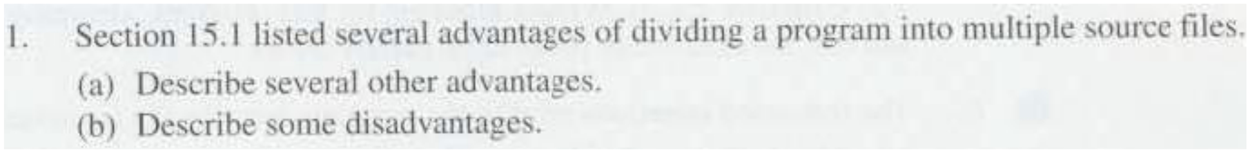
Some preprocessors delete white-space characters at the beginning of a line, so your results may vary. Three lines will cause errors when the program is compiled. Two contain syntax errors:

```
int a[= 10], i, j, k, m;
i = (x,y) x-y(j, k);
```

The third refers to an undefined variable:

```
i = jk;
```

Chapter 15: Exercises



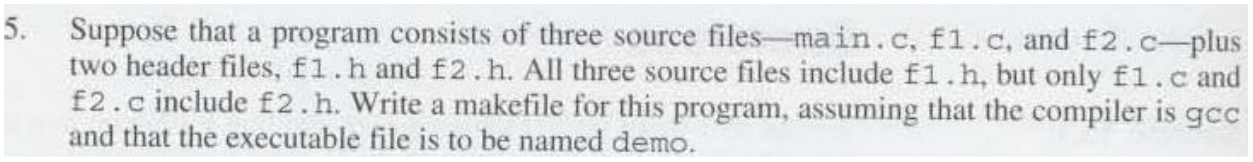
1. Section 15.1 listed several advantages of dividing a program into multiple source files.
   (a) Describe several other advantages.
   (b) Describe some disadvantages.

(a) Other advantages of dividing a program into multiple source files:

- Work can be divided among multiple programmers, with each programmer working independently on one or more files.
- Individual source files can be reviewed and tested independently of the others.
- Alternate versions of a source file can be created — to implement stubs or experiment with the relative efficiency of different algorithms, for example — and then linked to create alternate versions of the executable.
- Changes to individual files can be tracked at a more granular level, especially when used in conjunction with a version control system.

(b) Disadvantages of dividing a program into multiple source files:

- The use of multiple source files in a project, especially one that involves multiple programmers, requires that design be conducted more carefully before starting implementation. Although not a disadvantage in and of itself, design changes after implementation has started must be communicated and applied consistently across the project.
- Multiple source files increase the possibility of similar functions (especially "helper" functions) being developed and maintained independently of one another rather than being reused.
- An emphasis on reuse, without adequately grouping related functions into larger files, can result in a proliferation of source files.


5. Suppose that a program consists of three source files—`main.c`, `f1.c`, and `f2.c`—plus two header files, `f1.h` and `f2.h`. All three source files include `f1.h`, but only `f1.c` and `f2.c` include `f2.h`. Write a makefile for this program, assuming that the compiler is `gcc` and that the executable file is to be named `demo`.

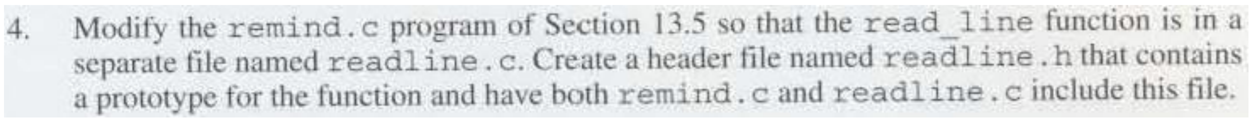
```
demo: main.o f1.o f2.o
        gcc -o demo main.o f1.o f2.o

main.o: main.c f1.h
        gcc -c main.c

f1.o: f1.c f1.h f2.h
        gcc -c f1.c

f2.o: f2.c f1.h f2.h
        gcc -c f2.c
```

Chapter 15: Programming Projects


4. Modify the `remind.c` program of Section 13.5 so that the `read_line` function is in a separate file named `readline.c`. Create a header file named `readline.h` that contains a prototype for the function and have both `remind.c` and `readline.c` include this file.

```
/* remind.c */

#include <stdio.h>
#include <string.h>
#include "readline.h"

#define MAX_REMIND 50   /* maximum number of reminders */
#define MSG_LEN 60      /* max length of reminder message */

int main(void)
{
  char reminders[MAX_REMIND][MSG_LEN+3];
  char day_str[3], msg_str[MSG_LEN+1];
  int day, i, j, num_remind = 0;
```

```c
  for (;;) {
    if (num_remind == MAX_REMIND) {
      printf("-- No space left --\n");
      break;
    }

    printf("Enter day and reminder: ");
    scanf("%2d", &day);
    if (day == 0)
      break;
    sprintf(day_str, "%2d", day);
    read_line(msg_str, MSG_LEN);

    for (i = 0; i < num_remind; i++)
      if (strcmp(day_str, reminders[i]) < 0)
        break;
    for (j = num_remind; j > i; j--)
      strcpy(reminders[j], reminders[j-1]);

    strcpy(reminders[i], day_str);
    strcat(reminders[i], msg_str);

    num_remind++;
  }

  printf("\nDay Reminder\n");
  for (i = 0; i < num_remind; i++)
    printf(" %s\n", reminders[i]);

  return 0;
}

/* readline.h */

#ifndef READLINE_H
#define READLINE_H

int read_line(char str[], int n);

#endif

/* readline.c */

#include <stdio.h>
#include "readline.h"

int read_line(char str[], int n)
{
  int ch, i = 0;

  while ((ch = getchar()) != '\n')
    if (i < n)
      str[i++] = ch;
  str[i] = '\0';
  return i;
}
```