York University
Electrical Engineering and Computer Science

EECS2031: Software Tools
SU2016
Assignment #6

Chapter 12: Exercises

1. Suppose that the following declarations are in effect:

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[1], *q = &a[5];
```

(a) What is the value of `*(p+3)`?
(b) What is the value of `*(q-3)`?
(c) What is the value of `q - p`?
(d) Is the condition `p < q` true or false?
(e) Is the condition `*p < *q` true or false?

(a) 14
(b) 34
(c) 4
(d) true
(e) false

4. Rewrite the `make_empty`, `is_empty`, and `is_full` functions of Section 10.2 to use the pointer variable `top_ptr` instead of the integer variable `top`.

```
int *top_ptr;

void make_empty(void)
{
  top_ptr = &contents[0];
}

bool is_empty(void)
{
  return top_ptr == &contents[0];
}

bool is_full(void)
{
  return top_ptr == &contents[STACK_SIZE];
}
```

5. Suppose that `a` is a one-dimensional array and `p` is a pointer variable. Assuming that the assignment `p = a` has just been performed, which of the following expressions are illegal because of mismatched types? Of the remaining expressions, which are true (have a nonzero value)?

(a) `p == a[0]`
(b) `p == &a[0]`
(c) `*p == a[0]`
(d) `p[0] == a[0]`

(a) Illegal
(b) Legal, true
(c) Legal, true
(d) Legal, true

6. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variable `i` and all uses of the `[]` operator.) Make as few changes as possible.

```
int sum_array(const int a[], int n)
{
  int i, sum;

  sum = 0;
  for (i = 0; i < n; i++)
    sum += a[i];
  return sum;
}
```

```
int sum_array(const int a[], int n)
{
  int *p, sum;

  sum = 0;
  for (p = a; p < a + n; p++)
    sum += *p;
  return sum;
}
```

17. Rewrite the following function to use pointer arithmetic instead of array subscripting. (In other words, eliminate the variables `i` and `j` and all uses of the `[]` operator.) Use a single loop instead of nested loops.

```
int sum_two_dimensional_array(const int a[][LEN], int n)
{
  int i, j, sum = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < LEN; j++)
      sum += a[i][j];

  return sum;
}
```

```
int sum_two_dimensional_array(const int a[][LEN], int n)
{
  int *p, sum = 0;

  for (p = a[0]; p < a[0] + n * LEN; p++)
    sum += *p;

  return sum;
}
```

Chapter 12: Programming Projects

1. (a) Write a program that reads a message, then prints the reversal of the message:

```
Enter a message: Don't get mad, get even.
Reversal is: .neve teg ,dam teg t'noD
```

Hint: Read the message one character at a time (using `getchar`) and store the characters in an array. Stop reading when the array is full or the character read is `'\n'`.

(b) Revise the program to use a pointer instead of an integer to keep track of the current position in the array.

(a)

```
#include <stdio.h>

#define MSG_LEN 80     /* maximum length of message */

int main(void)
{
  char msg[MSG_LEN];
  int i;

  printf("Enter a message: ");
  for (i = 0; i < MSG_LEN; i++) {
    msg[i] = getchar();
    if (msg[i] == '\n')
      break;
  }

  printf("Reversal is: ");
  for (i--; i >= 0; i--)
    putchar(msg[i]);
  putchar('\n');

  return 0;
}
```

(b)

```
#include <stdio.h>

#define MSG_LEN 80     /* maximum length of message */

int main(void)
{
```

```
  char msg[MSG_LEN], *p;

  printf("Enter a message: ");
  for (p = &msg[0]; p < &msg[MSG_LEN]; p++) {
    *p = getchar();
    if (*p == '\n')
      break;
  }

  printf("Reversal is: ");
  for (p--; p >= &msg[0]; p--)
    putchar(*p);
  putchar('\n');

  return 0;
}
```

3. Simplify Programming Project 1(b) by taking advantage of the fact that an array name can be used as a pointer.

```
#include <stdio.h>

#define MSG_LEN 80     /* maximum length of message */

int main(void)
{
  char msg[MSG_LEN], *p;

  printf("Enter a message: ");
  for (p = msg; p < msg + MSG_LEN; p++) {
    *p = getchar();
    if (*p == '\n')
      break;
  }

  printf("Reversal is: ");
  for (p--; p >= msg; p--)
    putchar(*p);
  putchar('\n');

  return 0;
}
```

4. Simplify Programming Project 2(b) by taking advantage of the fact that an array name can be used as a pointer.

```
#include <ctype.h>
#include <stdio.h>

#define MAX_MSG_LEN 80

int main(void)
{
  char msg[MAX_MSG_LEN], ch, *p, *q = msg;

  printf("Enter a message: ");
```

```
  while (q < msg + MAX_MSG_LEN) {
    if ((ch = getchar()) == '\n')
      break;
    if (isalpha(ch))
      *q++ = toupper(ch);
  }

  for (p = msg, q--; p < q; p++, q--)
    if (*p != *q)
      break;

  if (p >= q)
    printf("Palindrome\n");
  else
    printf("Not a palindrome\n");

  return 0;
}
```