ENEE 140 Lab 9

Lab instructions

This handout includes instructions for the recitation sessions on Wednesday and Friday. **Follow these instructions** to review to review the `switch`, `break` and `continue` statements, then **submit the homework** as indicated below.

1 Using `break` and `continue`

Modify the `break_continue.c` program from the class public directory on GRACE as follows:

1. Rewrite the two `for` loops with `while` loops, then with `do-while` loops.
2. What will be the output if you remove the first '`\n`' in the second and fourth `printf()` statements?
3. What will be the output if we use each of the following statement to replace the `for` loop

   `for (x=1;x<10; x++)`

   (a) `for (x=1; x<10; x=x+2)`

   (b) `for (x=2; x<10; x=x+2)`

   (c) `for (x=1; x!=10; x=x+2)`

   (d) `for (x=2; x!=10; x=x+2)`

2 `switch` practice

Rewrite the following code segment of `if-else` statements by using the `switch` statement. You can test your code by writing a complete C program. Make sure that you use representative values of a that cover all the possible cases.

```
if (a % 4 == 0) {
    printf("I am happy!\n");
} else {
    if (a % 4 == 2) {
        printf("I am not that happy.\n");
    } else {
        printf("I am sad. \n");
    }
}
```

3 Logical operators

Question 1: Verify De Morgan's laws by filling the missing values in the following truth table:

`expr1`	`expr2`	`!(expr1 && expr2)`	`!expr1 || !expr2`	`!(expr1 || expr2)`	`!expr1 && !expr2`
0	0				
0	1				
1	0				
1	1				

Question 2: Think about whether the following are true. If not, give a counter-example.

```
!(expr1 && expr2)  =  expr1 || expr2
(!expr1) && (!expr2) = !(expr1 || expr2)
```

Homework

Due: April 08 at 11:59 pm.

Create two programs by following the instructions below. Submit them using the following commands:

```
submit 2016 spring enee 140 AAAA 9 array_functions.c
submit 2016 spring enee 140 AAAA 9 array_reverse.c
```

Note: you must replace AAAA with your own section number (0101, 0102, etc.)

1 Functions for manipulating arrays

Write a complete program, called `array_functions.c`, that defines several functions to manipulate arrays. Implement the functions declared before `main()` to make the following program work:

```c
#include <stdio.h>

void printarray (int a[], int size);
// print out the content of array a[] with size integers
int even(int a[], int size);
// return the number of even numbers in array a[]
void insert(int a[], int size, int key, int k);
// insert key into array a[] so key becomes the k-th element
void second(int a[], int size);
// find the second largest number in a[] and swap it
// with the first element in the array
void rearrange(int a[], int size);
// rearrange array a[] so all the even numbers will
//  have indices less than those for the odd number

int main(void) {
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
  printarray (a, 12);
  printf("there are %2d even numbers.\n", even(a, 12));
  insert(a, 12, 20, 3);
  printarray (a, 12);
  insert(a, 12, 0, 0);
  printarray (a, 12);
  insert(a, 12, 30, 11);
  printarray (a, 12);
  second(a, 12);
  printf("calling second(): ...\n");
  printarray (a, 12);
  rearrange(a, 12);
  printf("calling rearrange(): ...\n");
  printarray (a, 12);
  return 0;
}
```

2 Array reverse

Write a program, called `array_reverse.c`, that reads an array from the terminal and that reverses the order of elements in the array. For example, `{2, 3, 5, 7, 11, 13, 17, 19}` will become `{19, 17, 13, 11, 7, 5, 3, 2}`. The program should write the array in the reverse order to the terminal.

Hint. Here's one idea for the implementation: swap `a[0]` and `a[n-1]`, `a[1]` and `a[n-2]`, etc. Note that if `n`=6, this swap must stop at the middle of the array, when you have swapped `a[2]` and `a[3]`. Think about what would happen if you keep on swapping until you reach the end of the array (when you are swapping `a[n-1]` and `a[0]`).

The following 3 statements swap the values of variables `a` and `b`:

```
int a, b, tmp;
tmp = a;      // make a copy of a's value in tmp
a = b;        // Overwrite a, as its original value is preserved in tmp
b = tmp;      // copy tmp (old value of a) to b
```

Reading assignment

K&R Chapters 5.10, 7.1, 7.5, 7.6, 7.7, B1. Re-read chapters 7.2, 7.4.

Weekly challenge

Write a program that reads several file names from the command line, concatenates these files, and prints the result to the standard output.

You can use the following template (also available in the GLUE class public directory, at `public/challenges/week09`):

```
/*
 * cat.c
 *
 *  Read several file names from the command line.
 *  Concatenate these files and print the result to the standard output.
 */

#include <stdio.h>


// Copy the content of one file to another; assume that the files are open.
// Return the number of characters copied
int
filecopy(FILE *in, FILE *out)
{
}

int
main(int argc, char *argv[])
{

        return 0;
}
```

```
/*
 * cat.c
 *
 *  Read several file names from the command line.
 *  Concatenate these files and print the result to the standard output.
 */

#include <stdio.h>


// Copy the content of one file to another; assume that the files are open.
// Return the number of characters copied
int
filecopy(FILE *in, FILE *out)
{
}

int
```

```
main(int argc, char *argv[])
{

	return 0;
}
```

The weekly challenge will not be graded. However, if you manage to solve it, you may submit it for extra credit. The deadline for submitting your solution to the weekly challenge is **Monday at 11:59 pm**. To be eligible for extra credit, you must implement correctly **all but two** of the weekly challenges. You can submit your program from a GRACE machine using the following command (replace `AAAA` with your section number):

```
submit 2016 spring enee 140 AAAA 1009 cat.c
```