DALHOUSIE UNIVERSITY
Inspiring Minds

Fall 2017 (Sep5-Dec5)
Faculty of Computer Science

CSCI 2132 — Software Development

Assignment 6

Due: *Thursday, Nov 16, 2017 by 11:59 p.m.*
Worth: 65 marks
Instructor: Vlado Keselj, CS bldg 432, 494-2893, vlado@dnlp.ca

---

Note: This is a partial A6 assignment. Please check later for a full version.

Assignment Instructions:

Solutions to this assignment must be submitted through SVN, in a similar way as for the previous assignment.

The first question refers to the Lab 7 and must be submitted in the SVN directory *CSID*`/lab7`, where *CSID* is your FCS userid.

The answers to all other questions must be submitted in your SVN directory: *CSID*`/a6` where *CSID* is your CS userid. Remember that you need to add to `svn` the directory as well as any files that you want submitted.

1) (10 marks) Make sure that you complete the Lab 7 as required. There are a set of files that need to be submitted in this lab in your SVN directory for the course named "*CSID*`/lab7`" where *CSID* is your CS userid.

You must make sure that the location of this directory is exactly as specified, and that the directory and all required files are submitted properly to SVN before the deadline.

The following files need to be submitted: `hello.c` (2 marks), `numbers.c` (2 marks), and `binary.c` (6 marks).

2) (20 marks) Record your ansswers in a plain textual file named `a6q2.txt` and submit using SVN.

a) (10 marks) Briefly explain the following function, and illustrate its input and output on a small example with $n = 5$:

```
int f1(int n, int a[]) {
  int *p = a, tmp = a[n-1];
```

```
  for (; n > 0; n--, p++) {
    int tmp1 = *p;
    *p = tmp;
    tmp = tmp1;
  }

  return n;
}
```

b) (10 marks) Briefly explain the purpose of the following function. Use a small example to illustrate the function.

```
int f2(char *s) {
  char *p;

  for (p = s; *p != '\0'; p++)
    ;
  for (--p; p >= s && (*p < '0' || *p > '9'); p--)
    ;

  if (p < s)
    return 1;

  while (p >= s && *p >= '0' && *p <= '9') {
    if (*p == '9')
      *p = '0';
    else {
      (*p)++;
      return 0;
    }
    p--;
  }

  if (p >= s && *p == ' ') {
    *p = '1';
    return 0;
  }

  return 1;
}
```

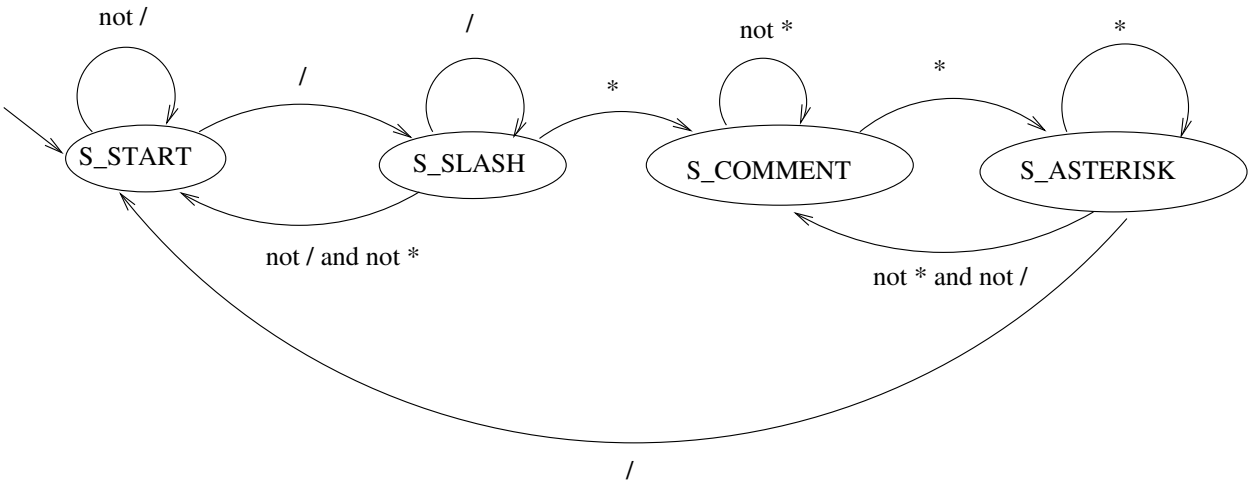
3) (20 marks) Write a C program named `a6q3.c` which removes C-style comments of the form "`/*...*/`" from the input.

The program must read standard input and produce standard output. Any character outside of a comment should be copied as it is. It is recommended that you use functions `getchar` and `putchar` for this copying. The comments must be skipped in copying with exception of the new-line characters. If any new-line characters are included in a comment, they must be copied to the output. This is done in order to preserve line positions of the text and the total number of lines from the input to the output.

The program must not store text unnecessarily, meaning that it must print characters as soon as possible after reading them. Sometimes it may need to wait with a character before printing; for example, after reading a slash character it needs to wait if the next character is asterisk or not to decide whether the comment is starting or not. Remember that you need to handle only the comments of the form "`/*...*/`", but not line comments starting with "`//`".

The input may end while the program is reading a comment, in which case, you should print the following line at the end:
`<COMMENT NOT FINISHED>`
ending with a newline character.

It is recommended that you use a finite automaton in processing, such as the following:



You can control the state of your program using the above diagram. You still need to decide what characters to output in each state as the program reads the input.

Sample Input:

The sample input file is provided as the file `a6q3-test.in` in the assignment URL and in the `~prof2132/public` directory on bluenose.

```
This is some text.
This is first /* comment */ etc.
```

```
Multiline comment: /* first line
second line
third line */ third line continued
Fourth line
A few tricks: ////*********comment***/ etc.
How about this one: /*/ this is still comment /*/ out
A comment can start /* and not
finish, and this should also count as a comment,
but print a warning!
```

Sample Output:

The sample output file is provided as the file `a6q3-test.out` in the assignment URL and in the `~prof2132/public` directory on bluenose.

```
This is some text.
This is first  etc.
Multiline comment:

 third line continued
Fourth line
A few tricks: /// etc.
How about this one:  out
A comment can start


<COMMENT NOT FINISHED>
```

4) (15 marks) Write a C function named `partition` following the quick sort algorithm for partition discussed in class (Lecture 20) and submit it in the file named `a6q4.c`

The function must work with the program `a6q4-main.c` which is available at the assignment web page and in the `~prof2132/public/` directory. You can notice that the function must work with pointers. The pivot must be chosen as the middle element, as shows in class, however, the code for getting this middle element is somewhat different than when using arrays and indices.

If you correctly implemented the function, then you can compile the program using the command:
`gcc -o a6q4-main a6q4-main.c a6q4.c`
and test it on the given test file with:
`./a6q4-main < a6q4-test1.in > a6q4-test1.new`
The produced output should be the same as the given file `a6q4-test1.out`.

If you want to test further that partition is done exactly as expected you can use another file that produced a more detailed trace (`a6q4-trace.c`) using the command:
`gcc -o a6q4-trace a6q4-trace.c a6q4.c`
and test it on the given test file with:
`./a6q4-trace < a6q4-test1.in > a6q4-test1-trace.new`
The produced output should be the same as the given file `a6q4-test1-trace.out`.

All given files can be found at the assignment web page or in the directory `~prof2132/public` on bluenose, and they are given here as well:

The file `a6q4-main.c`:

```
/* Program: a6q4-main.c Author: Vlado Keselj */
#include <stdio.h>

void quicksort(int *lo, int *hi);
int* partition(int *lo, int *hi);

int main() {
  int n;

  scanf("%d", &n);
  if (n > 0) {
    int a[n], i;
    for (i=0; i<n; i++)
      scanf("%d", &a[i]);

    quicksort(a, a+n-1);

    for (i=0; i<n; i++)
      printf("%d\n", a[i]);

    return 0;
  }
  else
    return 1;
}

void quicksort(int *lo, int *hi) {
  if (lo < hi) {
    int *p = partition(lo, hi);
    quicksort(lo, p);
    quicksort(p+1, hi);
  }
```

```
}
```

The file `a6q4-test1.in`:

```
10
-5
10 20 -1 0 -1
56 -25 10 10
```

The file `a6q4-test1.out`:

```
-25
-5
-1
-1
0
10
10
10
20
56
```

The file `a6q4-trace.c`:

```
/* Program: a6q4-trace.c Author: Vlado Keselj */
#include <stdio.h>

void quicksort(int *lo, int *hi);
int* partition(int *lo, int *hi);

int main() {
  int n;

  scanf("%d", &n);
  if (n > 0) {
    int a[n], i;
    for (i=0; i<n; i++)
      scanf("%d", &a[i]);

    quicksort(a, a+n-1);

    for (i=0; i<n; i++)
      printf("%d\n", a[i]);
```

```
    return 0;
  }
  else
    return 1;
}

void quicksort(int *lo, int *hi) {
  if (lo < hi) {
    int *p, *q;

    printf("PARTITION  IN:");
    for(q=lo; q<=hi; q++)
      printf(" %2d", *q);
    putchar('\n');

    p = partition(lo, hi);

    printf("PARTITION OUT:");
    for(q=lo; q<=hi; q++)
      printf("%c%2d", q==p ? '*' : ' ', *q);
    putchar('\n');

    quicksort(lo, p);
    quicksort(p+1, hi);
  }
}
```

The file `a6q4-test1-trace.out`:

```
PARTITION  IN: -5 10 20 -1  0 -1 56 -25 10 10
PARTITION OUT: -5 -25 -1 -1* 0 20 56 10 10 10
PARTITION  IN: -5 -25 -1 -1  0
PARTITION OUT: -5 -25*-1 -1  0
PARTITION  IN: -5 -25 -1
PARTITION OUT:*-25 -5 -1
PARTITION  IN: -5 -1
PARTITION OUT:*-5 -1
PARTITION  IN: -1  0
PARTITION OUT:*-1  0
PARTITION  IN: 20 56 10 10 10
PARTITION OUT: 10 10*10 56 20
PARTITION  IN: 10 10 10
```

```
PARTITION OUT: 10*10 10
PARTITION  IN: 10 10
PARTITION OUT:*10 10
PARTITION  IN: 56 20
PARTITION OUT:*20 56
-25
-5
-1
-1
0
10
10
10
20
56
```