BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

Second Semester 2017-18 CS F111 Computer Programming

LABORATORY SESSION #6 SOLUTIONS

(Control Flow)

1. Given below is an implementation:

```
#include <stdio.h>
int main()
{
char ch;
int no_vowels = 0, no_consonants = 0;
while (1)
{
  ch = getchar();
  if (ch == '\n' || ch == ' ' || ch == EOF)
        break;    /* word delimiter or end of input encountered */

  putchar(ch);    /* displaying each character so the entire word is displayed */

  if (ch < 'A' || (ch > 'Z' && ch < 'a') || ch > 'z') continue;
                  /* ignore the character if it is a non-alphabet */
  switch (ch)
  {
        case 'A': case 'a': case 'E': case 'e': case 'I': case 'i':
        case 'O': case 'o': case 'U': case 'u': ++no_vowels; break;
                  /* "fall through" if it is a vowel, upper case or lower case */
        default: ++no_consonants; break;
  }
}
printf(" has %d vowels and %d consonants.\n", no_vowels, no_consonants);
return 0;
}
```

- The crux of the program is the switch case with multiple options falling through.
- Notice the loop terminating conditions correctly.
- You may have also not paid attention to the fact that a non-alphabet character is to be ignored – note the elegant use of continue in this context.

2. Given while loop is:

```
while (1)
     printf("  TRUE FOREVER  ");
```

The equivalent for loop is:

```
for (    ;    ; printf(" TRUE FOREVER "))
              ;
```

Remember the ^C character to stop the infinite loop.

3. The core logic is to count the number of '\n' characters for counting number of lines, and every character in the file for giving the total count. You may use arrays, but this problem could be solved easily using character input with `getchar()`.

   a. `wc < sample.txt` is the command to be used, after the file is created (Point of emphasis: input redirection).

```
char ch;
int no_lines = 0, no_words = 0, no_chars = 0;

while ((ch = getchar()) != EOF)    /* You familiar with EOF */
{
  no_chars++;
  if (ch == '\n') no_lines++;
  if (ch == ' ' || ch == '\n' || ch == '\t') no_words++; /* though not part
            of the question, you may want to implement word count also */
}
printf("  %d  %d  %d\n", no_lines, no_words, no_chars);
```

b. Yes, they should match exactly.

c. You may use a sequence of two commands:
`who > no_users.txt ; ./my_wc < no_users.txt`
or use piping: `who | my_wc`

4.

a. A `while` loop or a `do … while` loop should be incorporated.

b. If the recent hailstone, say k, falls within the range [a, b) given by the user the control has to be transferred out the loop – using a `break` statement. The code would read something like this: `if (k >= a && k < b) break;`

5. The standard library function `rand()` is used to generate pseudo random numbers.

a. The same random numbers are generated in this case.

b. Now the random numbers generated vary each time the program is run.

c. To restrict it within [20, 40), this operation will do the job: `(rand() % 20) + 20;` Point of emphasis – the `%` operator restricts the value within [0, n), where n is the right operand of the operator. That is, `numb % x` will limit the answer in the range 0 … x-1 (both limits inclusive).

6. (a) Compile-time error – duplicate case value not allowed.

(b) Compile-time error – switch quantity has to be in integer; case labels have to reduce to integer constants.

(c) 12

21