CSCI 340 **Computer Assignment 5** **Fall 2017**

(10 points)

The Josephus Problem

The problem is known as the Josephus problem and postulates a group of soldiers of size N >= 1 surrounded by an overwhelming enemy force. There is no hope for victory without reinforcements, but there is only a single horse available for escape. The soldiers agree to a pact to determine which of them is to escape and summon help. They form a circle and a number M >= 1 is picked from a hat. One of their names is also picked from a hat. Beginning with the soldier whose name is picked, they begin to count clockwise around the circle. When the count reaches M, that soldier is removed from the circle, and the count begins again with the next soldier. The process continues so that each time the count reaches M, another soldier is removed from the circle. Any soldier removed from the circle is no longer counted. The last soldier remaining is to take the horse and escape. The problem is, given a number M, the ordering of the soldiers in the circle, and the soldier from whom the count begins, to determine the order in which soldiers are eliminated from the circle and which soldier escapes.

For example, suppose that M = 3 and there are N = 5 soldiers named A, B, C, D, and E. We count three soldiers starting at A, so that C is eliminated first. We then begin at D and count D, E, and back to A, so that A is eliminated next. Then we count B, D, and E, and finally B, D, and B, so that D is the one who escapes.

For this computer assignment, you are to write and implement a C++ program to simulate and solve the Josephus problem. The input to the program is the number M and a list of N names, which is clockwise ordering of the circle, beginning with the soldier from whom the count starts. After each removal, the program should print the name tags of all soldiers in the circle until only one soldier remains. However, to save the printing space, print the name tags for the remaining soldiers after only K elimination where K >= 1 is also an input to the program. The input arguments N, M, and K can be entered from stdin in the given order.

Programming Notes:

1. Name the soldiers in a circle in the following sequence: A1, A2, ..., A9, B1, B2, ..., B9, C1, C2, ... Enter the input arguments when the program prompts for them, and use a vector < string > to store the name tags of N soldiers.

2. In addition to the main ( ) routine, implement the following subroutines in your program:

   - void init_vals (vector < string >& v, inargs& in) : It reads the input values of N, M, and K of the struct object in when the program prompts for them and prints out those values on stdout. You can find the definition of the struct inargs in the header file prog5.h, which is defined as struct inargs { unsigned N, M, K; }; This routine also changes the size of the vector v to N and fills in the name tags for all soldiers in v.

- void print_vector (const vector < string >& v, const unsigned& cnt) : It prints out the contents of the vector v at the beginning and after removing K name tags from the vector until only one name tag remains, where cnt has an initial value of 0 and it simply indicates the total number of removals so far. At the end, it also prints the name tag of the last remaining soldier. For printout, print only up to 12 name tags in a single line, where the name tags are separated by single spaces.

3. The main ( ) routine first calls init_vals ( ) and initializes cnt to 0, and then calls the print_vector ( ) to print out the names of soldiers in circle. After that it locates the M-th name tag in the vector, and using the member function erase ( ), it removes that name tag from the vector, and calling print_vector ( ) to print out the contents of the vector. This process continues (in a loop) until only one name tag remains in the vector. Note: (a) If i (with the initial value 0) indicates the position of the soldier in the vector, then the statement (i + M – 1) % v.size ( ) returns the position of the M–th soldier from the starting position i. (b) Since the input argument to the erase ( ) function is an iterator, you can convert an index value i to an iterator by the statement: v.begin ( ) + i.

4. To store the name tags in an empty vector, first change the size of the vector to N, and then use the function generate ( ) in the STL. The last argument of this function is the function object SEQ (N), which is defined in the header file prog5.h. To use the header file in your program, insert the following statement at the top of your source file: #include "/home/cs340/progs/17f/p5/prog5.h".

5. To compile and link your program with the system library routines, execute: Make N=5. To test your program, execute: Make execute N=5. This command executes your program with the data in the input file prog5.d and displays the output as well as any error messages both on the terminal screen and in prog5.out. After you are done with your program, you don't need its object and executable files any more. To delete them, execute: Make clean. The input file prog5.d contains the test values N = 41, M = 3, and K = 7, but to test your program with arbitrary values of N, M, and K, execute: prog5.exe and enter the input values from the stdin.

6. You can find the correct output of this program in file prog5.out in directory: ~cs340/progs/17f/p5.

For your program, a proper documentation is required, no global variables are permitted, and for I/O operations, you're not allowed to use the functions from the C library.

When your program is ready, submit its source file to your TA, by executing: mail_prog.340 prog5.cc.