University of Victoria Final Examination December 2017

Last Name:	
First Name:	
UVic Student Number:	**V00**

Course Name & Number	CSC 111: Fundamentals of Programming with Engineering Applications
Section	A01, A02, A03, A04
CRN	13781, 13782, 10730, 10731
Instructor	Dr. H. A. Müller
Duration	3 hours

Instructions

- This examination consists of 12 pages including this cover page. Count the number of pages and report any discrepancy immediately to an invigilator.
- Answer the questions on the examination paper.
- There are 19 questions. The points are listed in square brackets at the beginning of each question.
- The last page of the exam contains a reference sheet for the C standard library.
- You have 3 hours for this examination. **Time management: approx. 5-7 minutes per question.**
- This examination is closed-books, closed-notes, no calculators, no gadgets, no cell phones, and no electronic devices. **Cell phones must be in your bag and must be turned off.**
- Turn in your completed final exam at the front of the examination room.
- For multiple choice questions, mark all circles that are next to correct choices.
- Be sure to complete the information on the declaration attached to this examination **including your signature.** Do not detach the declaration.
- **You are not permitted to leave before 15:30 p.m.**

1. [8] Assume the following declarations. Then allocate four different entities of storage on the ***heap*** using `malloc()`, which is defined in `<stdlib.h>`, as follows: *(1) 60 bytes, (2) one double basic type, (3) one **Node** data type, and (4) one **Complex** data type.* Note: The ***Node*** and ***Complex*** types are defined below. *C type casts* are required.

```
#include <stdlib.h>
typedef struct NodeStruct* NodeRef;
typedef struct NodeStruct {
    NodeRef next;
    NodeRef prev;
    int item;
} Node;
typedef struct {
    double re;
    double im;
} Complex;
```

2. [6] Write a syntactically correct C function **arithSeq(int start, int inc, int count)** that outputs an arithmetic sequence to the Console, where **start** is the first value in the sequence, **diff** is the difference between two consecutive values in the sequence, and **count** is the number of sequence elements to be generated.

```
void arithSeq(int start, int diff, int count) {
```

```
}/*arithSeq*/
```

3. [8] For each cell in the table below, answer the appropriate question with "**Yes**" or "**No**" for the respective word on the left—and the C programming language. Refer to the C standard library reference at the end of this exam.

	Is it a valid identifier?	**Is it a built-in C type?**	**Is it a reserved word (or keyword)?**	**Is it a standard library function?**
`printf`	Yes	No	No	Yes
`abc`				
`break`				
`string`				
`boolean`				
`strncat`				
`sizeof`				
`for`				

4. [6] Assume the following syntactically correct C type and variable declarations. Initialize **mary** with the date **December 12, 2017**. Then initialize **bob** using its **bp** pointer with the date **July 1, 1867**.

```
typedef struct {
    int year;
    int month;
    int day;
} Date;
Date mary;
Date bob;
Date* bp = &bob;
```

5. [4] What is the output of the following syntactically correct C program?

```
#include <stdio.h>
#include <stdlib.h>
#define ASIZE 8
void swap(int A[], int j, int k){
    int tmp = A[j];    A[j] = A[k];    A[k] = tmp;
}/*swap*/
int main(void) {
    int A[ASIZE] = {116, 110, 111, 115, 106, 225, 226, 265};
    for(int k = 0; k < ASIZE-1; k++){
        swap(A, k, k+1);
    }/*for*/
    printf("Result: %d %d\n", A[1], A[ASIZE-1]);
    return EXIT_SUCCESS;
}/*main*/
```

Output:

6. [4] What is the output of the following syntactically correct C program?

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   int k = 9;
   while (k<10) {
       printf("%d ", k);
       k = k - 1;
   }/*while*/
   printf("\n");
   return EXIT_SUCCESS;
}/*main*/
```

Output:

7. [4] What is the output of the following syntactically correct C program?

```
#include <stdio.h>
#include <stdlib.h>
void f4() { printf("Santa Claus "); }
void f3() { printf("is coming "); f4(); }
void f2() { printf("to town "); f4(); }
void f1() { printf("Rudolph "); f2(); }
int main(void){
    f1();
    f3();
    printf("reindeer\n");
    return EXIT_SUCCESS;
}/*main*/
```

Output:

8. [6] What is the output of the following syntactically correct C program?

```
#include <stdio.h>
#include <stdlib.h>
void pointerQuiz( int* p, int* q, int** R, int** T ){
    **R = 10;     *q = 4;
    *T = p;       **T = 6;
    *p = 5;       R = T;
    *R = p;       **T = 6;
}/*pointerQuiz*/
int main(void){
    int x = 9;          int y = 8;
    int* A = &x;        int* B = &y;
    int** C = &A;       int** D = &B;
    printf("%d %d %d ## ", *B, **C, **D);
    pointerQuiz( A, B, C, D );
    printf("%d %d %d\n", *B, **C, **D);
    return EXIT_SUCCESS;
}/*main*/
```

Output:

9. [4] Assume the following syntactically correct C declarations. Evaluate the expressions and compute the values of the variables **a**, **b**, **c** and **d**. Note for **a** and **b**, the required answer is either **true** or **false** (i.e., not 0 or 1).

a:

b:

c:

d:

```
#include <stdbool.h>
//...
int x = 9;
int y = 4;
int p = 9;
int q = 8;
int r = 20;
bool a = (y % x == y);
bool b = !((x + y) > p || (p + q) > r);
int c = (3*y + (x = 19)) % 7;
int d = (p + q) / y;
```

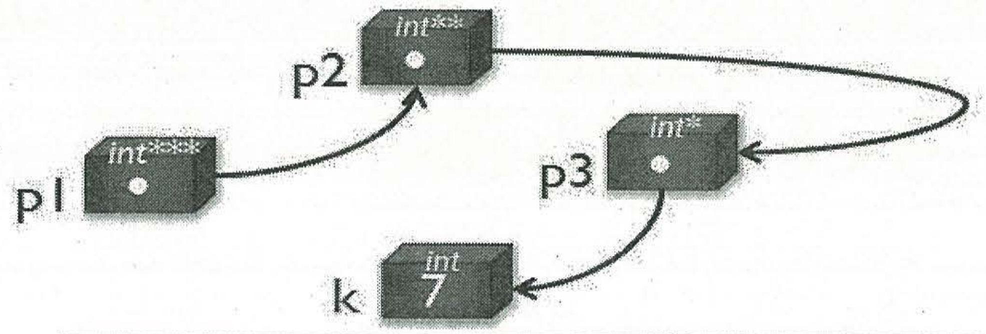
10. [2] Which statement is incorrect?

- ○ The main operations of a stack are **push()** and **pop()**.
- ○ The main operations of a queue are **enqueue()** and **dequeue()**.
- ○ The operations of a **stack** and a **deque** are subsets of the operations of doubly linked list.
- ○ A **queue** provides operations to insert and delete at the same end of the list.

11. [2] Which statement is incorrect?
    - ◯ A singly linked list can be traversed in one direction.
    - ◯ A doubly linked list can be traversed in both directions.
    - ◯ It is easier to insert and delete nodes into a doubly linked list than into a singly linked list.
    - ◯ The last element can be obtained faster in a doubly linked list than in a singly linked list.

12. [4] Describe the effect of expressions, such as `p = p->next` or `p = p->prev`, when **p** is pointing to a **`list element in a doubly linked circular list`**.

13. [6] Realize the following memory configuration using C variable declarations and pointer assignments. Note **k** is initialized to hold the value **7**. Then store the values **1**, **2** and **3** in **k** using pointers **p1**, **p2**, and **p3**, respectively. Of course, subsequent assignments will overwrite the value in **k**. Thus, at the end **k** will hold the value **3**.



14. [6] Complete the following C function **printLowercase()** so that it

  i. outputs all the lower case characters stored in array **A** on the console;
  ii. counts the number of lower case letters in array **A**;
  iii. outputs **LEN: 30; LC: 15** where 30 and 15 are the number of characters and lowercase letters stored in array **A**, respectively.

You may only use the **strlen()** function defined in **<string.h>** and the **islower()** function defined in **<ctype.h>**. Don't assign any values to the array or create additional arrays.

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

void printLowercase(char A[]) {
```

```
    printf("\n");
    printf("LEN: %d LC: %d\n", len, lc);
} /*printLowercase*/

int main(void) {
    printLowercase("Once Upon A Time There Was ...");
    return EXIT_SUCCESS;
} /*main*/
```

Here is the output produced by the following function call:

```
printLowercase("Once Upon A Time There Was ###");
nceponimehereas
LEN: 30; LC: 15
```

15. [8] Consider the partially written, syntactically correct C program below.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 6

void shiftArrayLeft(int A[], int len){
// Note the array A is already initialized with len values
```

```
}/*shiftArrayLeft*/

void printArray(int A[], int len){
// Note the array A is already initialized with len values
```

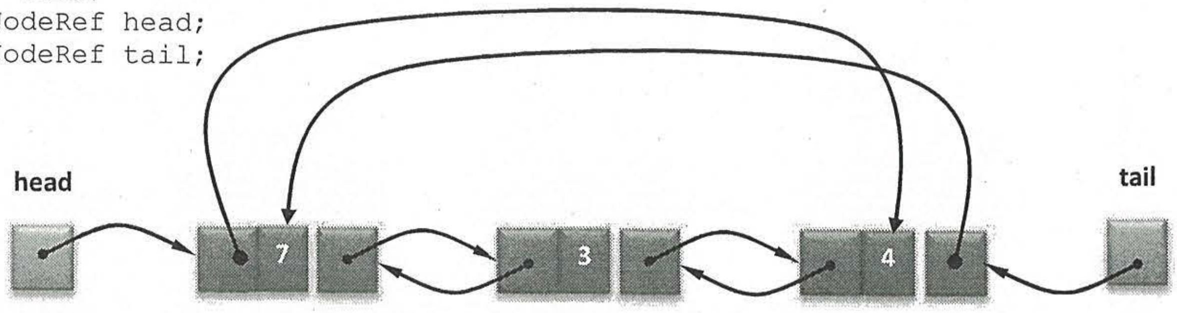
```
}/*printArray*/

int main(void){
    int A[6] = {19, 37, 44, 68, 72, 88};
    printArray(A, SIZE);
    shiftArrayLeft(A, SIZE);
    printArray(A, SIZE);
    return EXIT_SUCCESS;
}/*main*/
```

Implement the bodies of the functions **shiftArrayLeft()** and **printArray()** using **for** loops. The function **shiftArrayLeft()** is intended to shift the contents of the array provided as its argument by 1 position to the left. In the space provided above, complete the **shiftArrayLeft**() function such that the entire program compiles and produces the following output:

```
19, 37, 44, 68, 72, 88
37, 44, 68, 72, 88, 19
```

16. Assume that the **circular doubly linked list structure** depicted below was created using the following type and variable declarations.

```
typedef struct NodeStruct* NodeRef;
typedef struct NodeStruct {
    int item;
    NodeRef next;
    NodeRef prev;
} Node;
NodeRef head;
NodeRef tail;
```



a) [4] Give two different ways of changing the value of `item 3` to a value of `4` (i.e., write two different assignment statements) in the above circular doubly linked list. Note you don't have direct access to an individual list element. You must gain access to list elements starting with **head** or **tail**.

b) [4] Explain the effect of the following assignment statement applied to the above circular doubly linked list.

```
tail->next->next->prev->next->prev->prev->prev->next->item->info = 6;
```

17. [6] Describe in your own words the difference between **call-by-value** and **call-by-reference** parameter passing in the programming language C? Describe situations where call-by-value and call-by-reference parameter passing specifically apply. Be specific.

18. [2] Write a syntactically correct C expression that generates a random number in the **`range 31 to 47 (inclusive)`** using **`rand()`**, which is defined in **`<stdlib.h>`**. You do not need to write a complete C program or a function. No loops either. *Only one C expression.*

19. [6] Write a syntactically correct C **main()** function to generate 50 random numbers in the range of 44 to 88. Use the function **genRand()** below that uses the standard library function **rand()** to generate a single random number. Output every other generated random number with its counter (i.e., output 25 lines) to the Console each on a separate line as follows:

```
Rand 1 = 66
Rand 3 = 59
Rand 5 = 77
Rand 7 = 63
...
```

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>

    int genRand(int base, int range) {
        return rand() % range + base;
    } /*genRand*/

    int main(void) {
        srand(time(NULL));
```

```
        return EXIT_SUCCESS
    }/*main*/
```

***** END OF EXAMINATION *****

C Standard Library Reference (simplified)

Character Operations (ctype.h)

```
bool isalpha(char c);
bool isdigit(char c);
bool islower(char c);
bool isupper(char c);
char tolower(char c);
char toupper(char c);
```

General Functionality (stdlib.h)

```
void exit(int status);
void free(void* ptr);
void malloc(unsigned int size);
int rand();
void srand(unsigned int seed);
```

Input and Output (stdio.h)

```
int fclose(FILE* f);
int feof(FILE* f);
int fgetc(FILE* f);
char* fgets(char* buf, int bufsize, FILE* f);
FILE* fopen(char* filename, char* mode);
int fputc(int character, FILE* f);
int fputs(char* str, FILE* f);
int fprintf(FILE* f, char* format, ...);
int fscanf(FILE* f, char* format, ...);
```

Mathematics (math.h)

```
double cos(double x);
double exp(double x);
double log(double x);
double tan(double x);
double sin(double x);
double sqrt(double x);
```

Strings (string.h)

```
char* strcat(char* destination, char* source);
char* strncat(char* destination, char* source, int size);
int strcmp(char* str1, char* str2);
int strncmp(char* str1, char* str2, int size);
char* strcpy(char* destination, char* source);
char* strncpy(char* destination, char* source, int size);
int strlen(char* str);
int strnlen(char* str1, int maxsize);
```