University of Victoria Final Examination December 2015

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

Course Name & Number	CSC 111: Fundamentals of Programming with Engineering Applications	
Section	A01	A02
CRN	10736	10737
Instructor	H. A. Müller	B. Bird
Duration	3 hours	

Instructions

- This examination consists of 14 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 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 10:30 a.m.**

1. [3] In your own words, describe one reason to use call-by-reference for parameter passing instead of call-by-value.

2. [3] In your own words, describe one reason to use call-by-value for parameter passing instead of call-by-reference.

3. [4] 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. Cells left blank are assumed to be "No" answers. 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			Yes
`y`		No	No	No
`while`				
`float`				
`string`				
`strcmp`				
`c++`				

4. [2] Which of the following expressions generates a random number in the range 6 to 10 (inclusive)?

- ◯ `rand()%10 + 6`
- ◯ `rand()%5 + 6`
- ◯ `rand()%(10 - 6) + 5`
- ◯ `rand()%10 - 6`

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

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
     int k;
     k = 0;
     while( k < 27 ){
          printf("%d ",k);
          if ( k < 20 ){
               k = k + 15;
          }else{
               k = k + 7;
          }
     }
     while( k > 0 ){
          printf("%d ",k);
          k = k - 10;
     }
     printf("\n");
     return EXIT_SUCCESS;
} /* main */
```

Output:

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

```
#include <stdio.h>
#include <stdlib.h>
#define ASIZE 6
void swap(int A[], int j, int k){
     int tmp = A[j];
     A[j] = A[k];
     A[k] = tmp;
}
int main(void) {
     int A[ASIZE] = {106, 111, 116, 205, 225, 265};
     int i;
     for(i = 0; i < ASIZE-1; i++){
          swap( A, i, i+1 );
     }
     printf("Result: %d %d\n", A[0], A[ASIZE-1]);
     return EXIT_SUCCESS;
} /*main*/
```

Output:

7. [6] Consider the partially written, syntactically correct C program below.

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

#define MAX_NAME_LENGTH 100
```

```
int main(void){
     Student s;
     strcpy(s.name, "Polar Bear");
     s.student_number = 106205;
     s.gpa = 6.13;
     printf("Student %s (V00%06d): GPA = %.1f\n", s.name,
                 s.student_number, s.gpa);
     return EXIT_SUCCESS;
} /* main */
```

The code above will not compile, since the type `Student` is not defined. In the space provided above, write a syntactically correct definition of the `Student` type such that the entire program compiles and produces the following output:

```
Student Polar Bear (V00106205): GPA = 6.1
```

8. [8] Assume the following syntactically correct C declarations. Evaluate the expressions and compute the values of the variables a, b, c and d.

a:

b:

c:

d:

```
#include <stdbool.h>
//...
int x = 7;
int y = 1;
int p = 6;
int q = 10;
int r = 111;
bool a = (y % x == y);
bool b = !((x + y) < p || (p + q) > r);
int c = 3*y + (x = 19);
float d = (q + 2.0)/p;
```

9. [6] Consider the partially written, syntactically correct C program below. Refer to the C standard library reference at the end of this exam.

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

int count_uppercase(char s[]){
```

```
} /* count_uppercase */

int main(void){
     char s1[] = "CSC 111";
     char s2[] = "Blue Ridge Mountains";
     char s3[] = "Shenandoah River";
     printf("Uppercase letters in \"%s\": %d\n",
                           s1, count_uppercase(s1));
     printf("Uppercase letters in \"%s\": %d\n",
                           s2, count_uppercase(s2));
     printf("Uppercase letters in \"%s\": %d\n",
                           s3, count_uppercase(s3));
     return EXIT_SUCCESS;
} /* main */
```

The code above will not compile, since the function `count_uppercase` is not defined. In the space provided above, write a syntactically correct function called `count_uppercase` such that the entire program compiles and produces the following output:

```
Uppercase letters in "CSC 111": 3
Uppercase letters in "Blue Ridge Mountains": 3
Uppercase letters in "Shenandoah River": 2
```

10. [2] What is the difference between a `stack` and a `queue` data structure?

11. [6] Consider the partially written, syntactically correct C program below.

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

bool linearSearch(int A[], int length, int search_value){

} /* linearSearch */

int main(void) {
     int A[6] = {106, 111, 116, 205, 225, 265};
     int B[3] = {10, 6, 1};
     if (linearSearch(A, 6, 111)){
          printf("Value 111 found in A\n");
     }else{
          printf("Value 111 not found in A\n");
     }
     if (linearSearch(B, 3, 106)){
          printf("Value 106 found in B\n");
     }else{
          printf("Value 106 not found in B\n");
     }
     return EXIT_SUCCESS;
} /*main*/
```

The code above will not compile, since the function `linearSearch` is not defined. The `linearSearch` function should take an array `A`, its length and a value to search for in the array and return `true` if the value is found in the `array` and `false` otherwise. In the space provided above, write a syntactically correct function called `linearSearch` such that the entire program compiles and produces the following output:

```
Value 111 found in A
Value 106 not found in B
```

12. [2] Suppose the elements 10, 6, 15, 20 are inserted onto a stack data structure with the `push` operation. What is the value returned by a single `pop` operation?

13. This question covers file I/O and has three parts. All three parts use a text file called `cities.txt`, which contains seven lines of city names. The contents of `cities.txt` are listed below. You may assume that there are no spaces at the beginning or end of each line in the file.

```
Port Hardy
Victoria
Nanaimo
Comox
Duncan
Vancouver
Powell River
```

(a) [6] What is the console output of the following syntactically correct C program?

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

#define FILENAME ("cities.txt")
#define LINE_MAX 100

int main(void){
      char line[LINE_MAX];
      FILE* inputFile = fopen(FILENAME,"r");
      if (inputFile == NULL){
            fprintf(stderr,"Unable to open input file.\n");
            return EXIT_FAILURE;
      }
      while(fgets(line, LINE_MAX, inputFile) != NULL){
            printf("%c", line[0]);
      }
      printf("\n");
      fclose(inputFile);
      return EXIT_SUCCESS;
} /* main */
```

Output:

(b) [2] Consider the following lines from the code in part (a).

```
      if (inputFile == NULL){
            fprintf(stderr, "Unable to open input file.\n");
            return EXIT_FAILURE;
      }
```

This `if`-statement causes the program to exit if the value of `inputFile`, returned from `fopen`, is `NULL`. Give an example of circumstances in which `fopen` would return `NULL` in the program in part (a).

(c) [6] Consider the partially written, syntactically correct C program below. Refer to the C standard library reference at the end of this exam..

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define FILENAME ("cities.txt")
#define LINE_MAX 100

int count_lines(FILE* inputFile){
```

```
} /* count_lines */

int main(void){
      FILE* inputFile = fopen(FILENAME,"r");
      printf("Number of lines: %d\n", count_lines(inputFile) );
      fclose(inputFile);
      return EXIT_SUCCESS;
} /* main */
```

The code above will not compile, since the function `count_lines` is not defined. In the space provided above, write a syntactically correct function called `count_lines` which counts the number of lines of text in the provided file. On the `cities.txt` file, the complete program will produce the output

```
Number of lines: 7
```

14. [4] What is the output of the syntactically correct C program below?

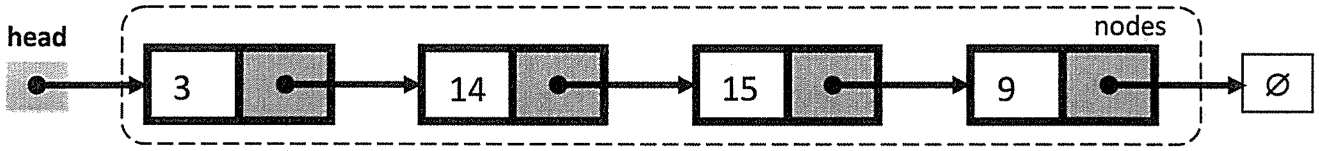
```
#include <stdio.h>
#include <stdlib.h>
void f4() { printf("f4 "); }
void f3() {
      printf("f3 ");
      f4();
}
void f2() {
      printf("f2 ");
      f4();
}
void f1() {
      printf("f1 ");
      f2();
}
int main(void){
      f1();
      f3();
      printf("Done\n");
      return EXIT_SUCCESS;
} /* main */
```

Output:

15. Assume that the singly linked list structure depicted below was created using the following type and variable declarations.

```
typedef int ListItem;
typedef struct ListNode  {
     ListItem value;
     struct ListNode* next;
} ListNode;

ListNode* head;
```



(a) [2] Write a single assignment statement to change the `value` of the node containing 15 to the value 6.

(b) [6] The code below creates a new `ListNode` object, which is referenced by the pointer `newNode`. Add syntactically correct C statements to in the space provided such that the new node is given the value `10` and is added to the beginning of the list (that is, before the value `3`).

```
ListNode* newNode = (ListNode*)malloc(sizeof(ListNode));
```

(c) [2] In your own words, describe one application where a linked list is a better choice than an array for storing a sequence of values.

16. [6] Consider the partially written, syntactically correct C program below.

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

typedef int Matrix[4][4];
typedef int Vector[4];

void print_matrix(Matrix M, int num_rows, int num_cols){

} /* print_matrix */
int main(void){
     Matrix M1 = { {1, 2, 3, 2},
                   {2, 3, 1, 4},
                   {0, 0, 0, 1},
                   {0, 1, 2, 3} };
     Matrix M2 = { {10, 20, 30, 0},
                   {40, 50, 60, 0},
                   { 0,  0,  0, 0},
                   { 0,  0,  0, 0} };
     printf("Matrix 1\n");
     print_matrix(M1, 4, 4);
     printf("Matrix 2\n");
     print_matrix(M2, 2, 3);

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

The function `print_matrix`, which outputs the provided matrix in tabular form, is not implemented. In the space provided above, complete the `print_matrix` function such that the entire program compiles and produces the following output:

```
Matrix 1
1 2 3 2
2 3 1 4
0 0 0 1
0 1 2 3
Matrix 2
10 20 30
40 50 60
```

17. [6] Consider the partially written, syntactically correct C program below.

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


void shiftArrayLeft(int A[], int length){
      int k;
      int tmp = A[0];
      for(k = 0; k < length-1; k++){
            A[k] = A[k+1];
      }
      A[length-1] = tmp;
}
void shiftArrayLeftByN(int A[], int length, int N){







} /* shiftArrayLeftByN */
void print_array(int A[], int length){
      int k;
      for(k = 0; k < length; k++){
            printf("%4d ", A[k]);
      }
      printf("\n");
}
int main(void){
      int A[6] = {6, 10, 106, 205, 111, 116};
      int B[5] = {3, 14, 15, 92, 65};

      shiftArrayLeftByN(A, 6, 3);
      print_array(A, 6);

      shiftArrayLeftByN(B, 5, 2);
      print_array(B, 5);

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

The `shiftArrayLeftByN` function should shift the contents of the array by N positions to the left. The provided function `shiftArrayLeft` shifts the contents of its argument by 1 position to the left. In the space provided above, complete the `shiftArrayLeftbyN` function such that the entire program compiles and produces the following output:

```
205  111  116    6   10  106
 15   92   65    3   14
```

18. [6] What is the output of the following syntactically correct C program?

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

void a_function(char S1[], char S2[], char output[]){
     int j, k;
     j = 0;
     k = 0;
     while(S1[j] != '\0' && S2[j] != '\0'){
          output[k] = S1[j];
          k++;
          output[k] = S2[j];
          k++;
          j++;
     } /* while */
     output[k] = '\0';
} /* a_function */

int main(void){
     char string1[] = "foe";
     char string2[] = "old";
     char string3[] = "shoe";
     char string4[] = "cold";

     char output[100];

     a_function(string1, string2, output);
     printf("%s %s: %s\n",string1, string2, output);

     a_function(string3, string4, output);
     printf("%s %s: %s\n",string3, string4, output);

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

Output:

19. [4] What is the output of the following syntactically correct C program? (Hint: draw a diagram)

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

void yet_another_function( int* p, int* q, int** R, int** T ){
     **R = 10;
     *p = 4;
     *T = p;
     **T = 6;
     *q = 5;
     R = T;
     *R = q;
     **T = 6;
} /* yet_another_function */

int main(void){
     int x = 0;
     int y = 0;
     int* A = &x;
     int* B = &y;
     int** C = &A;
     int** D = &B;
     printf("x = %d, y = %d\n", x, y);
     yet_another_function( A, B, C, D );
     printf("x = %d, y = %d\n", x, y);
     return EXIT_SUCCESS;
} /* main */
```

Output:

*** 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);
```