CSC 111 - FALL 2016
FINAL EXAMINATION
UNIVERSITY OF VICTORIA
December 5, 2016

Student Name:	
Student Number:	
Signature:	

Course Number	CSC 111	
Course Name	Fundamentals of Programming with Engineering Applications	
Section	A01	A02
CRN	10760	10761
Instructor	H. A. Müller	B. Bird
Duration	3 hours	

1. **This examination contains 21 questions on 14 pages, including this cover page. Please verify that your copy has all pages and notify the invigilator immediately if any pages are missing.**
2. **You have 3 hours to complete this exam.**
3. **Write all of your answers on this examination paper.**
4. **This exam is worth a total of 70 marks. The mark value of each question is given in square brackets at the beginning of the question.**
5. **No books, notes, calculators, phones, computers, smart watches or other electronic devices are permitted.**
6. **In multiple choice questions, clearly mark exactly one choice with an X.**
7. **You may not leave the examination until 20:30 (8:30pm), to allow your fellow students 90 minutes without interruptions.**
8. **If you have any backpacks or bags, leave them at the front of the room. The only items that should be at your desk are pens, pencils, the examination paper, your student ID card and possibly a snack/drink.**
9. **Place your student ID card face up on your desk.**
10. **Turn in your completed exam at the front of the room as you leave.**

1. [2 marks] What is the output of the syntactically correct C program below?

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   int k = 25;
   while (k < 50) {
      if (k%7 == 0) printf("%d ", k);
      k = k + 3;
   } /* while */
   printf("\n");
   return EXIT_SUCCESS;
}/* main */
```

Answer: ☐

2. [1 mark] Which of the following statements correctly initializes a pointer p to point to newly-allocated storage for 60 floating point values?

- ☐ `float p = (float)sizeof(float*)*malloc(60);`
- ☐ `float* p = (float)malloc(sizeof(float*)*60);`
- ☐ `float* p = (float*)malloc(60)*sizeof(float);`
- ☐ `float* p = (float*)malloc(sizeof(float)*60);`

3. [1 mark] Which of the following statements is correct?

- ☐ Nodes can be inserted at any point in a doubly linked list
- ☐ Doubly linked lists can only be traversed in one direction
- ☐ Linear search is always faster on doubly linked lists
- ☐ Nodes cannot be removed from singly linked lists

4. [3 marks] What is the output of the syntactically correct C program below?

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void){
   char s1[] = "Programming is cool";
   int len = strlen(s1);
   int k = len - 1;
   printf("\nWisdom = %s %d\n", s1, len);
   while (k >= 0) {
      printf("%c", s1[k]);
      k--;
   }/* while */
   printf(" = Wisdom %d\n", k);
   return EXIT_SUCCESS;
}/* main */
```

5. [2 marks] What is the output of the syntactically correct C program below?

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

void magic_wand(int* c, int d) {
   d = d + 6;
   *c = (*c)*2 - d;
   printf ("%d %d ", *c, d);
}/* magic_wand */

int main(void) {
   int x = 12, y = 9;
   magic_wand(&x, y);
   printf("%d %d\n", x, y);
   return EXIT_SUCCESS;
}/* main */
```

Answer:

6. [3 marks] Consider the partially written syntactically correct C code below.

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

double array_min_value(double A[], int length){
```

```
} /* array_min_value */

int main(){
    double A1[] = {3.14, 15.9, 2.65};
    printf("Min. element in A1: %.2f\n", array_min_value(A1, 3));
    double A2[] = {1.06, 1.16, 1.11};
    printf("Min. element in A2: %.2f\n", array_min_value(A2, 3));
    double A3[] = {3.71, 22.5, 2.30, 2.26};
    printf("Min. element in A3: %.2f\n", array_min_value(A3, 4));
    return EXIT_SUCCESS;
} /* main */
```

In the space above, write the definition of the array_min_value function, which takes an array A containing length values and returns the minimum value stored in the array. When your code is correct, the main() function above will produce the following output.

```
Min. element in A1: 2.65
Min. element in A2: 1.06
Min. element in A3: 2.26
```

7. [1 mark] Consider the assignment statement 'A[9][8][6] = 141;'. Which of the following declarations for the array A would be syntactically and semantically correct?

☐ `int A[10,9,7];`	☐ `int A[9][8][6][141];`
☐ `int A[15][4][2];`	☐ `int A[12][10][100];`

8. [4 marks] Consider the partially written syntactically correct C code below.

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

#define MATRIX_SIZE 5
typedef double Matrix[MATRIX_SIZE][MATRIX_SIZE];

void copy_matrix(Matrix source, Matrix destination, int num_rows, int num_cols){




} /* copy_matrix */
```

In the space above, write a syntactically correct definition of the `copy_matrix` function, which copies the contents of the matrix `source` (with `num_rows` rows and `num_cols` columns) into the matrix `destination`.

9. [1 mark] Which of the following statements is correct?

- ☐ An array groups elements of the same data type while a struct can group elements of varying types.
- ☐ There is no limit on the size of a struct, but arrays can have at most 65,536 elements.
- ☐ Arrays store only integers while structs store integer and string values
- ☐ A struct groups elements of the same data type while an array groups elements of different data types.

10. [4 marks] Consider the partially written syntactically correct C code below.

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

#define MATRIX_SIZE 5
typedef double Matrix[MATRIX_SIZE][MATRIX_SIZE];

int count_nonzero(Matrix M, int num_rows, int num_cols){
```

```
} /* count_nonzero */

int main(){
    int result;
    Matrix M1 = { {2, 0}, {1, 3} };
    Matrix M2 = { {1, 2, 3}, {0, 0, 0}, {106, 111, -116}, {2016, 5, 12} };
    Matrix M3 = { {3, 0, 14}, {15, 0, 9}, {0, 2, 6}, {53, 5, 0} };
    result = count_nonzero(M1, 2, 2);
    printf("First result: %d\n", result);
    result = count_nonzero(M2, 4, 3);
    printf("Second result: %d\n", result);
    result = count_nonzero(M3, 4, 3);
    printf("Third result: %d\n", result);
    return EXIT_SUCCESS;
} /*main*/
```

In the space above, write the definition of the count_nonzero function, which takes a Matrix M with dimensions num_rows × num_cols and returns the number of elements of M which are **not equal to zero**. When your code is correct, the main() function above will produce the following output.
Output:

```
First result: 3
Second result: 9
Third result: 8
```

11. [4 marks] Consider the partially written syntactically correct C code below.

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

void shift_string_right(char s[]){



} /* shift_string_right */

int main(){
    char s1[] = "earth";
    char s2[] = "electives";
    printf("First example: %s ", s1);
    shift_string_right(s1);
    printf("%s\n", s1);

    printf("Second example: %s ", s2);
    shift_string_right(s2);
    printf("%s\n", s2);
    return EXIT_SUCCESS;
} /* main */
```

In the space above, write the definition of the `shift_string_right` function, which takes a null-terminated C string and shifts each character one position to the right. The character at the end of the string is moved to the empty space created at the beginning. You may want to use one or more of the functions in `string.h` in your solution (documented in the reference sheet at the end of this exam). When your code is correct, the `main()` function above will produce the following output.

```
First example: earth heart
Second example: electives selective
```

12. Consider the syntactically correct C structure definition below.

```
typedef struct{
   int year;
   int month;
   int day;
} Date;
```

(a) [2 marks] In the space below, write the definition for the function `exam_date` (with the function signature given) which returns a `Date` struct containing **today's date** (i.e. 2016/12/05).

```
Date exam_date(){
```

```
}
```

(b) [2 marks] In the space below, write the definition for the function `print_date` (with the function signature given) which takes a pointer to a `Date` struct and prints out the year, month and day contained inside, using the format "year/month/day" (e.g. "2016/12/05").

```
void exam_date(Date* d){
```

```
}
```

13. [1 mark] Which of the following sequences of operators is ordered by operator precedence (from highest to lowest)?

- ☐ `[]  !  %=  <=`
- ☐ `()  <  &&  +=`
- ☐ `()  +  ==  ->`
- ☐ `()  ++  %  ||`

14. [6 marks] Consider the text file `desserts.txt` below. You may assume that there are no extra spaces at the beginning or end of each line of the file.

`desserts.txt`

```
Nanaimo Bar
Creme Brulee
Gelato
Tiramisu
Pumpkin Pie
Cheesecake
Apple Crisp
```

What is the output of the syntactically correct C program below?

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

#define FILENAME ("desserts.txt")
#define MAX_LINE 100

int main(void){
   FILE* ifp = fopen(FILENAME,"r");
   if (ifp == NULL){
      fprintf(stderr,"Unable to open input file.\n");
      return EXIT_FAILURE;
   }
   int k = 0;
   char line[MAX_LINE];
   while(fgets(line, MAX_LINE, ifp) != NULL){
      line[k+3] = '\0';
      if (k%6 != 5)
         printf("%s\n", line);
      k++;
   }
   return EXIT_SUCCESS;
}/* main */
```

15. [9 marks] Consider the syntactically correct C declarations below. Assume that the standard library header `stdbool.h` has been included.

```
int p = 12;
int q = 2016;
int* r = &q;
int* s = &p;
int** t = &s;
int** u = &r;
int Y[] = {17, 13, 12, 6};
bool Z[] = {true, false, false, true};
```

Give the results of the following syntactically correct C expressions, assuming the declarations above. Assume that the parts of the question are independent (and the result of one part does not affect the other parts).

a) `&p == r`

b) `u != &s`

c) `!(*r == 12)`

d) `*s <= **t`

e) `r != *u`

f) `(Y[0] != Y[1]) && (Y[2] == **t)`

g) `!((Y[1] <= Y[2]) && Z[0] && Z[1])`

h) `(Z[0] || Z[1]) && (Z[2] || Z[3])`

i) `(*s <= Y[0]) && !(*s <= Y[3])`

16. [3 marks] What is the output of the syntactically correct C program below?

```
#include <stdio.h>
#include <stdlib.h>
void mystery(char a[], char b[], int len) {
   int i = 0;
   while (i < len/2) {
      a[i+len/2] = a[i];
      a[i] = b[len/2+i];
      i++;
   }
}
int main(void){
   char s1[] = "ballot";
   char s2[] = "gopher";
   printf("%s %s ", s1, s2);
   mystery(s1, s2, 6);
   printf("%s", s1);
   return EXIT_SUCCESS;
}/* main */
```

Answer: ☐

17. [4 marks] For each of the questions below, mark the appropriate box for true or false.

TRUE	FALSE	
☐	☐	A single C array may contain elements with multiple types.
☐	☐	The total number of elements of an array in C must always be more than the number of dimensions.
☐	☐	The '\0' character must appear after the last character in a proper C string.
☐	☐	Arrays in C can have at most 3 dimensions.

18. [1 mark] Which of the following syntactically correct C expressions generates a random value between 111 and 116 (inclusive)?

☐ `rand()%(116 - 111) - 1` ☐ `111 + rand()%(116-111+1)`

☐ `rand()%116 - 111` ☐ `111 + rand()%(116-111) - 1`

19. [6 marks] In the space below, write a syntactically correct C program which opens the file "`input.txt`" for reading, opens the file "`output.txt`" for writing, copies the entire contents of the input file into the output file, then closes both files. Your code is not required to handle any error cases, including cases where the input file does not exist or where there is insufficient disk space to store the output file. You may want to consult the C standard library reference sheet at the end of the exam for a list of file I/O functions.

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

#define INPUT_FILE ("input.txt")
#define OUTPUT_FILE ("output.txt")

int main(){
```

```
} /* main */
```

20. [1 mark] The C preprocessor

☐ Translates C code to object code

☐ Checks for semantic errors

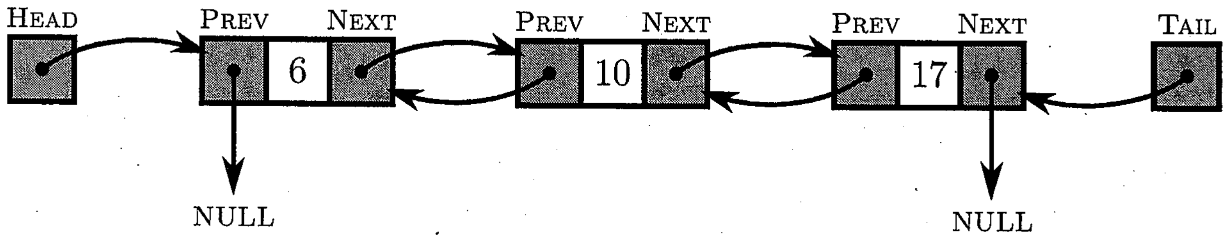
☐ Processes #include and #define directives

☐ Builds an application or executable from object code

21. Consider the syntactically correct C type definitions and declarations below.

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

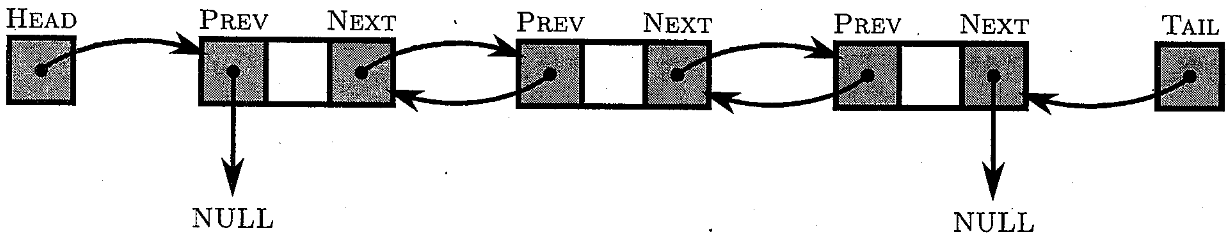
Suppose that the doubly linked list depicted below was created using the definitions and declarations above.



(a) [2 marks] Give **two different assignment statements** which both change the value 10 in the list above to the value 111.

(b) [3 marks] In the diagram below, write the element values for each node after running the following syntactically correct C statements on the list depicted above.

```
head->next->next->prev->element = 2;
tail->element = head->element;
head->element++;
head->next->element = head->next->next->element - 2;
```



*** 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);
```