Identifying information will be printed here.

University of Waterloo Midterm Examination CS 136

Term: Spring Year: 2016

Date:	June 20, 2016
Time:	7pm–8:50pm
Duration:	110 minutes
Sections:	001–03
Instructors:	Shoham Ben-David, Nam Pham

y87feng

Student Signature: ______________________

UW Student ID Number: ANSWER KEY

Number of Exam Pages (including this cover sheet)	21 pages
Exam Type	Closed Book
Additional Material Allowed	NO ADDITIONAL MATERIALS ALLOWED

This document is for the exclusive use of y87feng.

Instructions:

- **You may only use C language features discussed in Sections 01 through 07 of the course.**
- Your answers should always assume that code will be run in the Seashell environment.
- For each function you are asked to write, you are not required to provide any additional documentation unless we ask you to do so, but all helper functions must include a brief purpose statement.
- Your functions do not have to check for invalid parameters or assert requirements unless we ask you to do so.
- If you believe there is an error in the exam, notify a proctor. An announcement will be made if a significant error is found.
- It is your responsibility to properly interpret a question. **Do not ask questions regarding the interpretation of a question**; they will not be answered and you will only disrupt your neighbours. If there is a non-technical term you do not understand you may ask for a definition. If you are confused about a question, state your assumptions and proceed to the best of your abilities.
- You do not have to `#include` any of the standard libraries in your code (e.g., `stdio.h`) unless instructed to do so.
- For any code we provide you may assume that the necessary libraries has been included even if the code provided does not explicitly do so.
- If you require more space to answer a question, there are blank pages at the end you may use instead, but you must **clearly indicate** in the provided answer space that you have done so.
- **Do not detach any pages and do not write on the QR codes**
- For pointer parameters, you **must** use `const` when appropriate.

y87feng

Marking Scheme (for graders' use only)

Question	Max	Score	Grader
1	32		
2	12		
3	16		
4	10		

Maximum Total Score: 100

Question	Max	Score	Grader
5	10		
6	10		
7	10		

Total Score:

This document is for the exclusive use of y87feng.

Points on this page: 6

1. (32 points) Short Answers: Write concise, short answers. Point form is acceptable.

(a) (2 points) In Section 2 we described two levels of global scope. State the two levels and explain the significant difference between the two.

- The two levels of global scope are **program** and **module** scope
- Identifiers with program scope are accessible outside of the file / module where it is defined, while identifiers with module scope are only accessible inside of the module where it is defined

(b) (2 points) The functions `integer?` and `char?` are built-in to Racket. Explain why no equivalent functions exist in C.

In C all types are known in advance (static typing).

y87feng

(c) (2 points) Name two distinct reasons why the following expression might be false.
(assume `b != 0`):

```
a / b + a / b == (2 * a) / b
```

(Bonus point for naming three distinct reasons)

Integer arithmetic (`a/b rounded down`);
Overflow (`2 * a` is too big);
Floating-point arithmetic causes loss of precision.

This document is for the exclusive use of y87feng.

Points on this page: 7

(d) (3 points) Consider the following C code.

```
if (0 <= x <= 9) {
  printf("One digit number!\n");
}
```

This code always prints ”One digit number!” no matter what value x has. Explain why.

The first check will either be true (1) or false (0) and both of these values are less than or equal to 9.

Rewrite the boolean expression so it behaves as expected (is true if and only if x is between 0 and 9 inclusive).

```
if (0 <= x && x <= 9) ...
```

(e) (2 points) What does a return statement do? When is it valid to have a return statement that is not followed by an expression?

A return statement stops the function and returns to the line of code that called it. For a non-void function it requires an expression, and this expression determines the value of the function call expression. In a void function a return statement is not followed by an expression.

(f) (2 points) What is the value of an uninitialized variable?

If it is a global variable: 0. Otherwise: arbitrary.

y87feng

This document is for the exclusive use of y87feng.

Points on this page: 7

(g) (3 points) Consider the following definition

```
static int helper(int x, int y) {
  //...
}
```

What does the **static** keyword mean here?

It gives the function module scope.

How do you accomplish the same thing in a Racket module?

Don't use provide (by default globals have module, not program, scope).

(h) (2 points) What is integer overflow and why does it occur in C?

- Overflow is when you try to represent a value outside the range of possible values
- It occurs in C because integers are allocated a fixed number of bytes

(i) (2 points) For the following code:

```
int x = 42;
```

Write C code that mutates `x` using aliasing.

```
int *p = &x;
*p = 7;
```

This document is for the exclusive use of y87feng.

y87feng

Points on this page: 5

(j) (3 points) What is the minimum number of bytes of space that are reserved when the following is encountered?

```
struct mstruct {
   int x;
   int y;
   char c;
};
```

None. No space is reserved until a variable is created.

What is the minimum number of bytes of space that are reserved when the following is encountered?

```
struct mstruct  my_struct;
```

At least 9 bytes.

y87feng

(k) (2 points)

Consider the following C code.

```
struct posn *new_posn(int x, int y) {
  struct posn result = {x,y};
  return &result;
}
```

Explain what is wrong with this code.

After the function returns, the stack frame is removed from the call stack and the address of `result` is invalid.

This document is for the exclusive use of y87feng.

Points on this page: 7

(l) (2 points) Give two different reasons you might pass a variable's address as a parameter to a function.

- It allows the function to modify (mutate) the variable.
- It saves memory if the variable is a large structure (you only need to put a pointer on the stack, not the entire structure)
- It allows the function to (essentially) return multiple values and/or an extra error code

(m) (3 points) What does the following C program output?

```
#include <stdio.h>
int main(void) {
  int x = 10;
  int i = 0;
  for (x = 5; x < 10; ++i) {
    x += (2 * i);
    printf("%d\n", i/2);
  }
}
```

OUTPUT:

```
0
0
1
```

y87feng

(n) (2 points) Consider the following C code:

```
int *(*fn)(int, int *) = myfunc;
```

Give a declaration of `myfunc` so that the above code is correct.

int *myfunc(int a, int *b);

This document is for the exclusive use of y87feng.

Points on this page: 6

2. (12 points) Consider the following program:

```
struct posn { int x; int y; };

const struct posn my_posn = {0,2};

struct posn *max_min(const struct posn *a, const struct posn *b) {
  int a_min = a->x;
  if (a->y < a_min) {
    a_min = a->y;
  }
  // HERE
  int b_min = b->x;
  if (b->y < b_min) {
    b_min = b->y;
  }

  if (a_min > b_min) {
    return a;
  }
  return b;
}

int main(void) {
  struct posn p1 = {5,0};
  struct posn p2 = my_posn;
  struct posn *pp = max_min(&p1, &p2);
  printf("The largest is (%d %d)\n",pp->x,pp->y);
}
```

(a) (1 point) In the Seashell environment. what is the minimum number of bytes required to store the stack frame for `max_min`?

32 (16 for parameters, 8 for locals, 8 for return address)

(b) (4 points) For each of the following identifiers, indicate the area of the memory in which it is stored. Circle the appropriate answer.

`p2` :	**CODE**	**READ-ONLY DATA**	**GLOBAL DATA**	**STACK**
`my_posn` :	**CODE**	**READ-ONLY DATA**	**GLOBAL DATA**	**STACK**
`pp` :	**CODE**	**READ-ONLY DATA**	**GLOBAL DATA**	**STACK**
`max_min` :	**CODE**	**READ-ONLY DATA**	**GLOBAL DATA**	**STACK**

y87feng

This document is for the exclusive use of y87feng.

p2: Stack
my_posn: Read-Only
max_min: Code
pp: Stack

(c) (1 point) What is printed by the program?

```
The largest is (0 2)
```

y87feng

This document is for the exclusive use of y87feng.

Points on this page: 6

(d) (6 points)
Draw the call stack as it would appear when the program location reaches line 11.

Stack frames have the following format:

```
<Function_name>:
  <parameter1_name>: <parameter1_value>
  <parameter2_name>: <parameter2_value>
  ...
  <local_var1_name> [<addr_label1> (if needed)]:<local_var1_value>
  <local_var2_name>
     <field_name_1> : <field_value_1>
     <field_name_2> : <field_value_2>
  return addr: <function name> : <line number>
```

In a stack frame, the value of a pointer variable is either NULL or a label. A label has the format "addr_NUM" e.g. addr_1. It is only necessary to add a label to a variable if its address is stored in a pointer variable.

```
max_min:
   a : addr_1
   b : addr_2
   a_min : 0
   b_min : ???
   Return addr: main: 25 or 26

main:
   p1: [addr_1]
     .x : 5
     .y : 0
   P2: [addr_2]
     .x : 0
     .y : 2
   pp : ???

   Return Addr : to OS
```

y87feng

This document is for the exclusive use of y87feng.

3. (16 points) For this question, you will become the *Seashell* environment. The program below contains at least eight mistakes that won't let your code "run" (or would lead to unexpected results). Find eight (unique) problems with this code. *Note: poor style is not considered to be a mistake.*

For each unique problem, circle the problem code and/or draw an arrow at it, and number the problem (1,2,3...). In the table on the next page, briefly describe how you would *fix* each problem.

```
#include <rational.h>

struct rational {
    int num;
    int demom;
}

int gcd(int n, int m) {
    while (m != 0); {
        temp = n;
        n = m;
        m = temp % m;
    }
    return n;
}

struct rational simplify_rational(struct rational x) {
    int g = gcd(abs(x.num), abs(x.denom));

    rational r = {r.num / g, r.denom / g};
    if (r.denom < 0) {
        r = { -r.num, -r.denom };
    }

    return result;
}

int abs(int i) {
  return i > 0 ? i : -i;
}

bool compare_rationals (struct rational a, struct rational b) {
    return simplify_rational(a) == simplify_rational(b);
}

int main(void) {
    struct rational a = { 1, 2 };
    struct rational b = { 2, 4 };

    if (compare_rationals(a, b))
        printf("a and b are the same.\n");
    else
        printf("a and b are not the same\n.");
}
```

y87feng

This document is for the exclusive use of y87feng.

Points on this page: 16

Mistake	Line #	Brief Explanation & how you would fix it
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

y87feng

This document is for the exclusive use of y87feng.

Mistake	Line #	Brief Explanation & how you would fix it
1.	1	`#include <rational.h>` should be `#include "rational.h"`
2.	1/41	missing `#include <stdio.h>`
3.	1/32	missing `#include <stdbool.h>`
4.	6	missing semicolon after a struct declaration `struct rational {` `int num; int denom; };`
5.	9	infinite loop, remove the semicolon after `while (m != 0)`
6.	10	`temp` has not been declared `int temp = n;`
7.	20	`rational` should be `struct rational`
8.	20	`r.num(denom)` should be `x.num(denom)` (`r` has not yet been declared)
9.	22	not a valid assignment for structs, should be: `r.num = -r.num; r.denom = -r.denom;`
10.	25	`result` should be `r`
11.	33	`==` operator is not supported for structs, should be: `return r1.num == r2.num &&` `r1.denom == r2.denom;` `r1` and `r2` are simplified `a` and `b`
12.	28	abs defined too late
13.	5	`demom` should be `denom`

y87feng

This document is for the exclusive use of y87feng.

Points on this page: 10

4. (10 points) Consider the following C code snippet:

```
struct posn { int x; int y; };

struct posn my_posn = {7,13};
int main(void) {
  struct posn *p = &my_posn;
  int a = 17;
  int *b = &a;
  *b = 3;
  int *c = &(p->x);
  //Here
}
```

For each function declaration given below, circle all valid arguments that could be passed to the function with the given declaration at the point in the code marked by `//Here`.

(a) (2 points) `int *function1(int g);`

`a    &a    *a    b    &b    *b    c    &c    *c    p->x    &(p->x)`

a, *b, *c, p->x.

(b) (2 points) `int function2(int *g);`

`a    &a    *a    b    &b    *b    c    &c    *c    p->x    &(p->x)`

&a, b, c, &(p->x)

(c) (2 points) `int *function3(int **g);`

`a    &a    *a    b    &b    *b    c    &c    *c    p->x    &(p->x)`

&b, &c

For each function declaration given below, circle all variables that could be assigned the value returned by the function at the point in the code marked by `//Here`.
That is, for each function, if we assign `g = function( ... );`, what could `g` be?

(d) (2 points) `int function4(void);`

`a    &a    *a    b    &b    *b    c    &c    *c    p->x    &(p->x)`

a, *b, *c, p->x

y87feng

This document is for the exclusive use of y87feng.

(e) (2 points) `int *function5(void);`

```
a    &a    *a    b    &b    *b    c    &c    *c    p->x    &(p->x)
```

b, c

Important Comment:
On the exam, `my_posn` mistakenly appeared as a **const**:
`const struct posn my_posn = {7,13};`

If `my_posn` is a constant array, the code in this question does not compile, since the non-constant pointer `p` cannot be assigned the address of a constant structure.
If `p` was a **const** itself, then the answers to (b) and (d) above would be different.

In (b), `&(p->x)` would not be allowed.
In (d), `p->x` would not be allowed.

When marking, we accepted both versions of the answers.

y87feng

This document is for the exclusive use of y87feng.

5. (10 points) Write a C function `void word(void)` that uses `scanf` to read ASCII characters from the input (ignoring spaces) until two consecutive characters form a two-letter English word. Two English letters form a word if one of them is a vowel (one of `'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O'` or `'U'`) and the other is not. For example: `uP, da, AM, Li` are all words, but `UO` and `gh` are not. If a word is found, it prints it out and returns. If `scanf` fails to read a character, it returns without printing anything.

You may use the functions

```
  bool is_letter(char c);
  bool is_vowel(char c);
```

The above return `true` if `c` is an English letter / a vowel, respectively.

```
void word(void) {
  char c1, c2 ;
  if (scanf (" %c", &c1) != 1)
     return;
  while (1) {
    if (scanf (" %c", &c2) != 1)
      return;
    if ((is_letter(c1) && !is_vowel(c1) &&
          is_vowel(c2)) ||
          (is_letter(c2) &&
             !is_vowel(c2) && is_vowel(c1))) {
      printf("%c%c\n",c1,c2);
      return;
    }
    c1=c2;
  }
}
```

y87feng

This document is for the exclusive use of y87feng.

6. (10 points) Robotron Incorporated has hired you to write software for their robot. They have sent you the program below, noting by `// HERE` the place where you must add two function definitions. (See next page for more instructions).

```
#include <stdio.h>

const int NORTH = 0, WEST = 1, SOUTH = 2, EAST = 3;

struct posn {
  int x;
  int y;
};

struct robot {
  int dir;
  struct posn pos;
};

//  HERE

int main(void) {
  struct posn testbot_start_posn = {0,0};
  struct robot testbot = {NORTH, testbot_start_posn };

  turn_robot(&testbot, EAST);
  move_robot(&testbot, 3);
  turn_robot(&testbot, NORTH);
  turn_robot(&testbot, NORTH);
  move_robot(&testbot, 5);
  turn_robot(&testbot, 42);
  move_robot(&testbot, 1);
  turn_robot(&testbot, SOUTH);
  move_robot(&testbot, 2);
  turn_robot(&testbot, WEST);
  move_robot(&testbot, 1);
}
```

The output should be:

```
Robot has turned.
Robot moved to position (3,0).
Robot has turned.
Robot moved to position (3,5).
ERROR: INVALID DIRECTION
Robot moved to position (3,6).
Robot has turned.
Robot moved to position (3,4).
Robot has turned.
Robot moved to position (2,4).
```

This document is for the exclusive use of y87feng.

y87feng

Points on this page: 10

(6. Continued) Write the function definitions of `turn_robot` and `move_robot` in the space provided below. Each function should return `void`, and output text using `printf`. Your job is to ensure the program on the previous page prints the output at the bottom of the previous page.
Note that `turn_robot` prints ”Robot has turned.” only if the direction changes. `move_robot` accepts only positive integers.

```
void turn_robot(struct robot *r, int d) {
  if (d != r->dir) {
    if(0 <= d && d < 4) {
      r->dir = d;
      printf("Robot has turned.\n");
    } else {
      printf("ERROR: INVALID DIRECTION\n");
    }
  }
}

void move_robot(struct robot *r, int dist) {
  if (r->dir == NORTH) {
    r->pos.y += dist;
  } else if (r->dir == WEST) {
    r->pos.x -= dist;
  } else if (r->dir == SOUTH) {
    r->pos.y -= dist;
  } else if (r->dir == EAST) {
    r->pos.x += dist;
  }

  printf("Robot moved to position (%d,%d).\n",
    r->pos.x, r->pos.y);
}
```

y87feng

This document is for the exclusive use of y87feng.

Points on this page: 10

7. (10 points) Consider the C function `void staircase(int n);` which takes a positive integer `n` and prints according to the following examples:
You must use iteration, and cannot use recursion or helper functions.
`staircase(5);` should print

```
1
22
333
4444
55555
4444
333
22
1
```

`staircase(2);` should print

```
1
22
1
```

You may assume `n` will be 2 or greater.

```
void staircase(int n) {
       for (int i = 1; i <= n;i++) {
           for (int j = 0; j < i;j++) {
               printf("%d",i);
           }
           printf("\n");
       }
       for (int i = n-1; i >= 1;i--) {
           for (int j = 0; j < i;j++) {
               printf("%d",i);
           }
           printf("\n");
       }
}
```

y87feng

This document is for the exclusive use of y87feng.

This page is intentionally left blank for your use. Do not remove it from your booklet. If you require more space to answer a question, you may use this page, but you must **clearly indicate** in the provided answer space that you have done so.

y87feng

This document is for the exclusive use of y87feng.

This page is intentionally left blank for your use. Do not remove it from your booklet. If you require more space to answer a question, you may use this page, but you must **clearly indicate** in the provided answer space that you have done so.

y87feng

This document is for the exclusive use of y87feng.