Carleton University
SYSC 2006 A/B – Foundations of Imperative Programming – Fall 2017
Sample Midterm Exam

Name:_____

Student Number:_____

This exam contains questions from the Fall 2016 midterm exam.

INSTRUCTIONS:

1. This exam is closed book. Calculators are permitted.
2. **Exam questions will not be explained, and no hints will be given. If you think something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of your solution, and answer the question.**
3. Do not write anything on this page other than your name and student number.
4. Answer all questions on the question paper. You can use pen or pencil.
5. Do not detach any sheets from the question paper.
6. You are not permitted to use the calculator/clock apps on your cell phone. You can use a simple (non-programmable) calculator. The time will be written on the blackboard, and updated periodically throughout the exam.
7. An attendance sheet will be distributed after the exam begins. If you finish your exam early, do not leave until you have signed the attendance sheet.
8. If you need to go to the washroom during the exam, bring your question paper to the teaching console at the front of the classroom.
9. For all questions, assume that all required `#include` statements are in the source code.
10. For the multiple choice questions, circle the letter or letters that correspond to the correct answers. Some questions may have more than one correct answer. 1 mark will be awarded for each correct answer and the same number of marks will be deducted for each incorrect answer; however, the minimum mark you can receive for each question is 0.
11. The crib sheet at the end of this question paper summarizes some C functions that you may find useful.
12. When you have finished your exam, give it to one of the proctors. Show your campus card to the proctor, who will verify that the name and student number on your card match what is written on the question paper. You must leave the exam room as soon as you have turned in your paper.
13. All pages of this question paper must be turned in.

Question 1 [10 marks]

When this program runs, function `sum_pair` is called twice:

```
int sum_pair(int *m, int n)
{
    int sum = *m + n;    // Point A
    return sum;
}

int increment_and_sum(int *x, int *y)
{
    *x = *x + 2;
    *y = *y + 2;
    return sum_pair(x, *y);
}

int main(void)
{
    int a = 5;
    int b = 10;
    int first, second;
    first = sum_pair(&a, b);
    second = increment_and_sum(&a, &b);
    return 0;
}
```

(a) Draw a memory diagram similar to one that would be produced by the C Tutor, which depicts the program's activation frames immediately after the statement at Point A has been executed for the the **first** time, that is, immediately after the assignment statement,

`int sum = *m + n;`

is executed for the first time, but before the function returns.

Frames Objects

(b) Draw a memory diagram similar to one that would be produced by the C Tutor, which depicts the program's activation frames immediately after the statement at Point A has been executed for the the **second** time, that is, immediately after the assignment statement,

```
int sum = *m + n;
```

is executed for the second time, but before the function returns.

Frames Objects

Question 2

Which of the following are not legal names for C variables?

A. `fiveDogs`
B. `5Dogs`
C. `five_dogs`
D. `_5_dogs`
E. `*fivedogs`

Question 3

Which of the following statements about C functions is correct?

A. The body of a function must contain at least one `return` statement.
B. Assigning a value to a function parameter never changes the corresponding function argument.
C. A function call must contain at least one argument.
D. If a function has a `return` statement, any call to that function must appear on the right side of an assignment statement.
E. None of the statements are correct.

Question 4

A C program contains this function definition and other statements not shown here:

```
void silly(int a)
{
    printf("%d", a);

}
```

The length of time for which parameter `a` exists is:

A. Really long.
B. From the time the function is called until the function returns.
C. From the time the `main` function starts executing until `main` returns.
D. None of the above.

Question 5

A C program contains these two functions:

```
void fuzzy(int x)
{
    x = 73;
    printf("%d", x);
}

void bunny(void)
{
    int x = 29;
    fuzzy(x);
    printf("%d", x);

}  // This question continues on the next page.
```

What is displayed when `bunny` is called?

A.
```
29
29
```

B.
```
73
73
```

C.
```
29
73
```

D.
```
73
29
```

E.
```
29
```

F.
```
73
```

Question 6

Consider function `mystery`:

```
void mystery(int g, int f)
{
    g = g + 5;
    f = f + 9;
    printf("%d %d\n", f, g);
}
```

What numbers does `mystery` print when it is called by this code fragment?

```
int f = 2;
int g = 5;
mystery(f, g);
```

A. `10 11`
B. `11 10`
C. `7 14`
D. `14 7`
E. `None; instead, an error message is displayed.`

Question 7

Assuming that `point_t` is the `typedef`'d name of a `struct` "type", how many instances of the `struct` will we have when the following code is executed?

```
point_t *first_pt = malloc(sizeof(point_t));
point_t *second_pt = malloc(sizeof(point_t));
point_t *third_pt = second_pt;
```

A. `0`
B. `1`
C. `2`
D. `3`

Question 8

Read the following incomplete function and choose the best expression to fill in the blank on line 8 so that the function will behave correctly:

```
/*
* Takes an array of n integers and counts the number of times
* that 5 appears in the array.
*/
int count_fives(int arr[], int n)                    // line 1
{
    int counter = __________;                        // line 2
    int total_fives = 0;                             // line 3
    while (counter < __________) {                   // line 4
        if ( __________ == 5) {                      // line 5
            total_fives = total_fives + __________;  // line 6
        }
        counter += 1;                                // line 7
    }
    return __________;                               // line 8
}
```

A. counter
B. true
C. false
D. total_fives
E. arr

Question 9

Identify the single missing line of code:

```
int p, *r, *q;
p = 27;
r = &p;

// MISSING LINE
printf("The value is %d", *q); // Prints 27
```

A. *q = *r;
B. *q = r;
C. *q = &r;
D. q = *r;
E. q = r;
F. q = &r;
G. *q = *p;
H. *q = p;
I. *q = &p;
J. q = *p;
K. q = p;
L. q = &p;

Question 10

Function `find_min` contains a logic error on a single line in the function body, on one of the four lines indicated by comments:

```
/* This function returns the value of the minimum element in the
 * subsection of the array "y", starting at position "first" and
 * ending at position "last".
 */
int find_min(int[] y, int first, int last)
{
    int best_so_far = y[first];          // line 1

    for (int i = first + 1; i <= last; i++) {
        if (y[i] < y[best-so_far]) {     // line 2
            best_so_far = y[i];          // line 3
        }
    }
    return best_so_far;                  // line 4
}
```

Which one of the four lines indicated by the comments contains the logic error?

A. line 1
B. line 2
C. line 3
D. line 4

Question 11

What is output by this program?

```
int main(void)
{
    int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    for (int i = 0; i < 10; i += 2) {
        if (!(arr[i] % 2 == 0)) {
            printf("%d", arr[i]);
        }
    }
    return 0;
}
```

A. Nothing is printed
B. 02468
C. 13579
D. 01234
E. 56789
F. 0123456789

Question 12

Where is the logic error in the following implementation of a sequential search?

```
1  void sequential_search(int arr[], int n, int target)
2  {
3      _Bool target_found = false;
4      int target_location;
5      int current = 0;
6      while (current < n) {
7          if (arr[current] == target) {
8              target_found = true;
9              target_location = current;
10          } else
11              target_found = false;
12          current = current + 1;
13     }
14     if (target_found)
15         printf("target found at location %d\n", target_location);
16     else
17         printf("target not found\n");
18 }
```

A. Line 9
B. Line 6
C. Line 11
D. Line 7
E. Line 14
F. None of the above.

Question 13

What value will `i` contain after this code fragment is executed?

```
int *p, i;
i = 3;
p = &i;
(*p) += 1;
i += 1;
```

A. 3
B. 4
C. 5
D. 6
E. 7
F. 8

Question 14

Function `range_sum` is passed an array of `n` integers, plus the low and high end of a range of values within the array. The function calculates the sum of the values in the array that are within the range (but not including the range end values). Choose the best choice to fill in the blank on Line 7 so that the function will work as intended:

```
int range_sum(int arr[], int n, int low, int high)
{
    int num = 0;                                  // Line 1
    int sum = 0;                                  // Line 2

    for (int idx = 0; idx < n; idx += 1) {        // Line 3
        int val = arr[i];                         // Line 4
        if (__________) {                         // Line 5
            num += 1;                             // Line 6
            sum = __________;                     // Line 7
        }
    }
    return __________;                            // Line 8
}
```

A. `sum + 1`
B. `sum + n`
C. `sum + num`
D. `sum + val`
E. `sum + idx`

Question 15

Consider this code fragment:

```
int i, *q, *p;
p = &i;
q = p;
*p = 5;
// Insert call to printf here
```

Which of the following statements will output "`The value is 5`"?

A. `printf("The value is %d", &i);`
B. `printf("The value is %d", p);`
C. `printf("The value is %d", *q);`
D. `printf("The value is %d", *i);`

Question 16

A function contains the declaration, `int arr[50];` and the statement:

```
*(arr + 20) = 6;
```

Select the equivalent statement that uses the [ ] operator.

```
A. arr[21] = 6;
B. *arr[20] = 6;
C. arr[20] = 6;
D. *arr[21] = 6;
```

Question 17

This question refers to a function named `swap`, for which part of the code is shown below:

```
void swap(int x[], int i, int j)
{
    // swaps elements "i" and "j" of array "x".
    int temp;    // temporary storage for swapping

    xxx missing code goes here xxx
}
```

The missing code from `swap` is:

```
A.
temp = x[i];
x[j] = x[i];
x[j] = temp;

B.
temp = x[j];
x[j] = x[i];
x[j] = temp;

C.
temp = x[j];
x[i] = x[j];
x[j] = temp;

D.
temp = x[i];
x[i] = x[j];
x[j] = temp;

E.
temp = x[i];
x[j] = x[i];
x[i] = temp;
```

Question 18

Which statements will allocate an array that can store 10 integers, from the heap? Assume that `arr` is declared this way: `int *arr;`

A. `arr = malloc(sizeof(int)) * 10;`
B. `arr = malloc(sizeof(int * 10));`
C. `arr = malloc(sizeof(int) * 10);`
D.
```
for (int i = 0; i < 10; i += 1) {
    arr[i] = malloc(sizeof(int));
}
```

Question 19

What will be printed by this program?

```
void change_numbers(int numbers[], int n)
{
    for (int i = 0; i < n; i++)
        numbers[i] *= 2 ;
}

int main(void)
{
    int numbers[] = {1, 2, 3};
    int c = sizeof(numbers) / sizeof(numbers[0]);
    change_numbers(numbers, c);
    for (int i = 0; i < c; i += 1)
        printf("%d ", numbers[i]);

    return 0;
}
```

A. `1 2 3`
B. `2 2 3`
C. `2 4 3`
D. `2 4 6`
E. `2 2 2`
F. `0 0 0`

Question 20

What will this code fragment output?

```
int array[] = {45, 67, 89, 22, 13};
int *array_ptr = &array[1];
printf("%d, ", array_ptr[1]);
printf("%d, ", *(array_ptr+2));
array_ptr += 1;
printf("%d\n", array_ptr[1]);
```

A. 45, 67, 89
B. 45, 67, 67
C. 45, 67, 45
D. 67, 89, 22
E. 67, 89, 89
F. 67, 89, 67
G. 89, 22, 13
H. 89, 22, 22
I. 89, 22, 89

Question 21

What does this program output?

```
void example(int *n, int *m)
{
    *n += 1;
    *m += 5;
}

void main()
{
    int n = 2;
    int m = 5;
    example(&n, &m);
    printf("n = %d and m = %d", n, m);
}
```

A. n = 2 and m = 5
B. n = 3 and m = 5
C. n = 2 and m = 10
D. n = 3 and m = 10

Question 22

A program calls `malloc` to allocate an array from the heap, and stores the pointer returned by `malloc` in a variable named `arr`. After the program has finished using `arr`, which statements will free the array?

```
A. free(arr);
B. free(*arr);
C. free(&arr);
D. free(arr[0]);
E. free(*arr[0]);
F. free(&arr[0]);
G. free(arr[1]);
H. free(*arr[1]);
I. free(&arr[1]);
```

Question 23

Suppose you have the following incomplete function to sum the first `n` integers in array `nums`:

```
int sum_array(int nums[], int n)
{
    int sum = 0;
    ...
    return sum;
}
```

Which of the following code fragments <u>does not</u> correctly complete this function?

```
A.
for (int i = 0; i < n; i += 1) {
    sum += nums[i];
}
```

```
B.
for (int i = n - 1; i >= 0; i += 1) {
    sum += nums[i];
}
```

```
C.
int i = 0;
while (i < n) {
    sum += nums[i];
    i += 1;
}
```

Question 24

Consider this code fragment:

```
typedef struct {
  int num;
  int den;
} fraction_t;
...
fraction_t fr = {2, 3};
fraction_t *ptr = &fr;
```

What is the value of the expression, `*ptr`?

A. `2` (the value stored in member `num` in `fr`)
B. `3` (the value stored in member `den` in `fr`)
C. the entire structure `fr`
D. the address (location) of `fr`
E. the address (location) of member `num` in `fr`
F. the address (location) of member `den` in `fr`

Crib Sheet

`void *malloc(int size);`

Allocates *`size`* bytes from the heap and returns a pointer to the memory. The memory is not initialized. On error, the function returns `NULL`.

`void free(void *ptr);`

Frees the memory pointed to by `ptr`, which must have been returned by a previous call to `malloc`. Otherwise, or if `free(ptr)` has already been called, undefined behaviour occurs. If *`ptr`* is `NULL`, no operation is performed.