University of Maryland College Park  
Dept of Computer Science  
CMSC216 Spring 2016  
Midterm I Key

<table>
<thead>
<tr>
<th>#</th>
<th>Problem Description</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Problem #1 (C Constructs)</td>
<td>(80)</td>
</tr>
<tr>
<td>#2</td>
<td>Problem #2 (Memory Map)</td>
<td>(40)</td>
</tr>
<tr>
<td>#3</td>
<td>Problem #3 (Arrays, Strings/Coding)</td>
<td>(80)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>Total (200)</td>
<td><strong>(200)</strong></td>
</tr>
</tbody>
</table>

Grader Use Only
Problem #1

1. (4 pts) Write a Unix command that will copy the directory `project1` (and all its contents) that is in the current directory to the directory `all_projects` in your home directory.

   **Answer:**
   
   `cp -r project1 ~/all_projects`

2. (4 pts) Complete the following compilation command of the file `my.c` so the name of the executable is `my.exe`.

   `gcc`

   **Answer:**
   
   `gcc -o my.exe my.c`

3. (4 pts) When is the #define directive handled (analyzed and replaced with the provided value)?

   a. During translation
   b. During preprocessing
   c. During linking
   d. When the program is executed.

   **Answer:** b.

4. (4 pts) Which of the following relationships must always be true in C?

   a. char type can be 1 or 2 bytes; `sizeof(char) <= sizeof(int) < sizeof(long)`
   b. char type can be 1 or 2 bytes; `sizeof(char) <= sizeof(int) <= sizeof(long)`
   c. char type must be 1 byte; `sizeof(char) <= sizeof(int) <= sizeof(long)`
   d. None of the above

   **Answer:** c.

5. (4 pts) Circle those considered false in C.

   a. '0'
   b. 0
   c. NULL
   d. -1
   e. '0'
   f. ""

   **Answer:** a, b, c
6. (4 pts) Which of the following are valid prototypes for a function that returns a pointer to an integer and takes two parameters, a pointer to an integer and a double? Circle the valid prototypes.

   a. `int *process(int *x, double y);`
   b. `int *process(int *, double y);`
   c. `int *process(int *, double);`
   d. `void process(int *, double y);`
   e. `int process(int *, double y);`
   f. None of the above.

   **Answer: a, b, c**

7. (4 pts) A program consists of two files: main.c and help.c. In help.c there is a function that has the following prototype: `static int f();` Which of the alternatives below are true? (circle all that apply).

   a. The function `f()` can be called from main.c.
   b. There can be two functions named `f()` in the program; one in main.c and one in help.c.
   c. The static function `f()` can only call functions that are in help.c.
   d. Only static functions can call function `f()`.

   **Answer: b**

8. (4 pts) For the following code, which of the alternatives below are true (circle all that apply).

   ```c
   #include <stdio.h>

   int main() {
     double t = 101.5;
     double * const s = &t;
     double const *d = &t;

     return 0;
   }
   ```

   a. We can change `t` after it has been initialized.
   b. We can change `s` after it has been initialized.
   c. We can update `t` by assigning a value to `*s`.
   d. We can update `t` by assigning a value to `*d`.

   **Answer: a, c**
9. (5 pts) Indicate whether the following code compiles. If it does not compile indicate why; otherwise indicate what happens when the code is executed and the user enters the integer value 6.

```c
#include <stdio.h>

int main() {
    int val = 77;
    int *p = &val;
    int **m = &p;
    
    scanf("%d", *m);
    printf("%d
\n", val + 10);
    return 0;
}
```

**Answer:**
The code compiles. The output is:

16

10. (5 pts) The following code compiles and generates output. What is the output assuming an integer occupies 4 bytes, a character 1 byte, a double 8 bytes and pointers 8 bytes?

```c
#include <stdio.h>

void look(int d[]) {
    printf("V2: %ld %ld\n", sizeof(d), sizeof(d[0]));
}

int main() {
    int a[4] = {10, 20, 3, 9};
    
    printf("V1: %ld %ld\n", sizeof(a), sizeof(a[0]));
    look(a);
    return 0;
}
```

**Answer:**

V1: 16 4
V2: 8 4
11. (5 pts) Indicate whether the following code compiles. If it does not compile indicate why; otherwise indicate what happens when the code is executed.

```c
#include <stdio.h>

int main() {
    float y, w;
    float *m = &y;
    *m = w;
    printf("%f\n", y);
    return 0;
}
```

**Answer:**

Code compiles; a trash/garbage value is assigned to *m (y).

12. (5 pts) The following program relies on command line arguments. Below the % represents the Unix prompt.

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Processing 1\n");
    } else {
        printf("%s %s\n", argv[0], argv[1]);
    }
    return 0;
}
```

a. What is the output generated by the program when we execute the program as follows?

```bash
% a.out 2
```

**Answer:** Processing 1

b. What is the output generated by the program when we execute the program as follows?

```bash
% a.out 7 ab
```

**Answer:** a.out 7

13. (6 pts) Indicate whether the following code compiles. If it does not compile indicate why; otherwise indicate what happens when the code is executed.

```c
#include <stdio.h>

int main() {
    int i = 3, *ip = &i;
    void *vp = ip;
    printf("%d %d\n", *ip, *vp);
    return 0;
}
```

**Answer:** Code does not compile
14. (6 pts) Indicate whether the following code compiles. If it does not compile indicate why; otherwise indicate what happens when the code is executed.

```c
#include <stdio.h>

int main() {
    double a[3] = {1.0, 2.0, 3.0}, b[3];

    b = a;
    printf("%f %f %f\n", b[0], b[1], b[2]);
    return 0;
}

Answer: Code does not compile
```

15. (16 pts) Implement the function `strcpy()`, which copies the contents of a source C string into a destination C string and returns a pointer to the destination string. You can assume dest and src are not NULL and dest is large enough to hold the copied string.

```c
char *strcpy(char *dest, const char *src){

    One Possible Answer:

    char *strcpy(char *dest, const char *src) {
        int i = 0;
        while (src[i]) {
            dest[i] = src[i];
            i++;
        }
        dest[i] = '\0';
        return dest;
    }

    Another Possible (Really Short) Answer:

    char *strcpy(char *dest, const char *src) {
        char *p = dest;
        while (*p++ = *src++);
        return dest;
    }
```
Problem #2

Draw a memory map for the following program at the point in the program execution indicated by the comment */HERE */.

```c
#include <stdio.h>
#define MAX_LEN 5

void show(int *sp, int **f, int *m, int x) {
    *sp += 200;
    sp++;
    x += (m - sp);
    (*f)++;
    **f = 33;
    f = NULL;
    /* HERE */
}

int main() {
    int data[MAX_LEN] = {70, 5, 69, 2};
    int *p = data, *t = data + 4;
    int y = 3;
    show(data, &p, t, y);
    return 0;
}
```

Answer:

```
<table>
<thead>
<tr>
<th>f</th>
<th>NULL</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>sp</th>
<th>p</th>
<th>m</th>
<th>t</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data</th>
<th>270</th>
<th>33</th>
<th>69</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
</table>
```
Problem #3

Implement the function `reverse` that has the prototype below. The function initializes the out parameter `result` with a copy of the original string (`source`), a delimiter (separator parameter) and the reversed string. The following driver and associated output illustrates the functionality associated with the function you are expected to write.

```
Driver:
int main() {
    char result[81];
    char *p = reverse("abc", '*', result);
    printf("result: %s\n", result);
    printf("p: %s\n", p);
    return 0;
}
```

```
Output:
result: abc*cba
p: abc*cba
```

For this problem:

- From the string library you can only use the functions `strlen`, `strcpy` and `strcmp`. You will lose at least 60 pts if you use any other string library functions.
- The function returns a pointer to the string generated by the function.
- You can assume the array parameter (`result`) is large enough to store the result.
- You may not modify the `source` string parameter.
- The function will not perform any computation and return NULL if:
  - `source` is NULL
  - the length of the `source` string is 0
  - the `separator` character is the null character
- **Your code must be efficient**

```
char *reverse(const char *source, char separator, char result[])
```

**IMPLEMENT THE FUNCTION ON THE NEXT PAGE**
One Possible Answer:

```c
char *reverse(const char *source, char separator, char result[]) {
    size_t source_length;
    char *p = result;
    int i;

    if (source == NULL || separator == '\0') {
        return NULL;
    }
    if ((source_length = strlen(source)) == 0) {
        return NULL;
    }

    strcpy(result, source);
    p = result + source_length;
    *p++ = separator;
    for (i = source_length - 1; i >= 0; i--) {
        *p++ = source[i];
    }
    *p = '\0';

    return result;
}
```