Midterm Exam, Fall 2015 Date: October 29th, 2015

Instructions:

- This midterm exam takes 70 minutes. Read through all the problems and complete the easy ones first.
- This exam is OPEN BOOK. You may use any books or notes you like. However, the use of any electronic devices including laptops, ipads, phones etc. is forbidden.
1 Multiple choice questions (25 points):

Answer the following multiple-choice questions. Circle all answers that apply. Each problem is worth 5 points.

A. The 8-bit register %al has a hex value of 0xff, what could be its corresponding value in decimal?
   1. 128
   2. 255
   3. -1
   4. 256
   5. None of the above

B. Consider the assembly instruction, \texttt{movq (%rax), %rcx}. If %rax represents a C variable, what are its possible data types?
   1. char **
   2. char *
   3. int *
   4. long *
   5. int
   6. long
   7. None of the above

C. Suppose \texttt{int c = 0xdeadbeef}, which of the following statements clears the rightmost two bits in \texttt{c} and leaves the rest of the bits unchanged.
   1. \texttt{c = 0xdeadbeec}
   2. \texttt{c &= 0xffffffffc}
   3. \texttt{c = (c >> 2) \&\& 2}
   4. \texttt{c |= 0xffffffffc}
   5. \texttt{c |= 0x00000003}
   6. None of the above
D. What is the content of array A after executing the following code snippet?

```c
long A[3] = {1, 2, 3}
long *p;
long **q;
p = A;
p++;
q = &p;
p++;
(*p) = (**q) * 2;
```

1. 1 2 3  
2. 1 2 4  
3. 1 2 6  
4. 1 4 3  
5. 1 4 6  
6. None of the above

E. Consider the following code snippet,

```c
float f = 1.0;
int b = *(int *)&f;
int c = 1;
```

Which of the following logical statements are true?

1. b != c  
2. b > c  
3. b < c  
4. b == c  
5. None of the above
2 C basics (25 points):

Ben Bitdiddle wants to write a program that sorts a set of strings using bubble sort. Below is the main body of Ben’s program. It should print out three sorted strings, “apple orange pear”.

```c
1: void
2: main() {
3:  char strs[3][20] = {"orange", "apple", "pear"};
4:  for (int i = 0; i < 3; i++) {
5:    for (int j = i; j < 3; j++) {
6:      //strcmp returns positive if str[i] is greater than str[j] in alphabetical order
7:      if (strcmp(strs[i], strs[j]) > 0) {
8:        swap1(strs[i], strs[j]);
9:      }
10:    }
11:  }
12:  for (int i = 0; i < 3; i++)
13:    printf("%s ", strs[i]);
14: }
```

(a) (10 points) Please implement the `swap1` function to enable Ben’s program to sort correctly. You may want to use the standard library function such as `strlen`, `strcpy` (see Appendix I) and others.

```c
void
swap1(char *s1, char *s2) {
}
```
(b) (5 points) Ben has changed his code at line 3 to be \texttt{char \*strs[3] = \{"orange", "apple", "pear"\};}. Does his program that invokes \texttt{swap1} still work correctly? Please explain.

(c) (10 points) Ben has changed his code at line 8 to be \texttt{swap2(&strs[i], &strs[j])}. Please implement the \texttt{swap2} function so that Ben’s program will sort correctly when line 3 is changed to be \texttt{char \*strs[3] = \{"orange", "apple", "pear"\};}. 
3 Assembly (20 points):

Ben Bitdiddle has encountered a mystery function. Its machine code is given below.

0000000000000000 <mystery>:
  0: 48 89 f8 mov %rdi,%rax
  3: eb 09   jmp e <ex5+0xe>
  5: 48 39 30 cmp %rsi,(%rax)
  8: 74 09   je 13 <ex5+0x13>
 a: 48 8b 40 08 mov 0x8(%rax),%rax
 e: 48 85 c0 test %rax,%rax
11: 75 f2   jne 5 <ex5+0x5>
13: f3 c3   repz retq

Ben is given the hint that the first argument of the mystery function is of the type node *, where node is defined as follows:

typedef struct node {
  long value;
  struct node *next;
} node;

(a) (5 points) How many arguments does function mystery take? If there are multiple arguments, what are their types?

(b) (5 points) Does the function have a return value? If so, what is the type of its return value?
(c) (10 points) Complete the puzzle by filling in the blanks for the corresponding C implementation of mystery.

```c
mystery(node *head, __________ )
{
    node *n = head;

    while (____________________) {
        if (____________________) {
            break;
        }
    
        ________________;
    }

    return _______________; 
}
```
4 C and Buffer Overflow (30 points):

(a) (10 points) Implement a C function, called replace(), that replaces the first occurrence of a given substring s1 in str with another given substring s2. The function replace returns 1 if replacement has happened, otherwise it returns zero. For example, given the following code snippet, char str[20] = "hello"; int r = replace(str, "ll", "LLL"); The content of str should be "heLLLo" afterwards. And r should be 1.

For your implementation, you should just assume the argument str corresponds to a buffer large enough to hold the new string after replacement.

You may use standard library functions such as strstr or strcpy (the Appendix I contains the man pages for these two functions), but you do not have to.

//str, s1, s2 are null-terminated strings
int
replace(char *str, char *s1, char *s2) {
    //your code here
}
Consider the following code snippet that invokes `replace()`.

```c
int dangerous() {
    char buf[8] = "hello x";
    int r = replace(buf, "x", "friends and families");
    return r;
}

void main() {
    dangerous();
    printf("Nothing bad happened\n");
}
```

Its corresponding object code is shown below. Note the unfamiliar `movabs` instruction moves an 8 byte constant value into the destination register.

```
000000000040065d <replace>:
  40065d: 41 57 push %r15
  ...                      push %r15
  400726: c3 retq
0000000000400727 <dangerous>:
  400727: 48 83 ec 10 sub $0x10,%rsp
  40072b: 48 b8 68 65 6f 6c 6f movabs $0x78206f6c6f6568,%rax
  400732: 20 78 00
  400735: 48 89 04 24 mov %rax,(%rsp)
  400739: ba f4 07 40 00 mov $0x4007f4,%edx
  40073e: be fc 07 40 00 mov $0x4007fc,%esi
  400743: 48 89 e7 mov %rsp,%rdi
  400746: e8 12 ff ff ff callq 40065d <replace>
  40074b: 48 83 c4 10 add $0x10,%rsp
  40074f: c3 retq
0000000000400750 <main>:
  400750: 48 83 ec 08 sub $0x8,%rsp
  400754: b8 00 00 00 00 mov $0x0,%eax
  400759: e8 c9 ff ff ff callq 400727 <dangerous>
  40075e: bf fe 07 40 00 mov $0x4007fe,%edi
  400763: e8 b8 fd ff ff callq 400520 <puts@plt>
  400768: 48 83 c4 08 add $0x8,%rsp
  40076c: c3 retq
```
(b) (5 points) Read the machine code for function dangerous. If one is to print out the 8-byte constant 0x78206f6c6c6568 as an array of 8 ASCII characters, what are its contents? (If you need to consult an ASCII table, you may refer to Appendix II)

(c) (5 points) What is string stored at memory address 0x4007fc? What is the string stored at memory address 0x4007f4?

(d) (5 points) What is the value of the 8-bytes stored at the memory location (%rsp) just before executing the first instruction in dangerous (i.e. 400727: sub $0x10,%rsp)?

(e) (5 points) Running this code gives a segmentation fault. What is the last instruction executed before the segmentation fault occurs? Please circle one.

1. The retq instruction in main function, i.e. 40076c: retq.
2. The retq instruction in dangerous function, i.e. 40074f: retq.
3. The retq instruction in replace function, i.e. 400726: retq.
4. The instruction to deallocate from stack in dangerous, i.e. 400727: sub $0x10,%rsp.
Appendix I: strstr, strcpy, strncpy

STRSTR(3) Linux Programmer’s Manual

NAME
strstr - locate a substring

SYNOPSIS
#include <string.h>

char *strstr(const char *haystack, const char *needle);

DESCRIPTION
The strstr() function finds the first occurrence of the substring needle in the string haystack. The terminating null bytes ('\0') are not compared.

RETURN VALUE
The strstr() function returns a pointer to the beginning of the substring in haystack, or NULL if the substring is not found.

STRCPY(3) Linux Programmer’s Manual

NAME
strcpy, strncpy - copy a string

SYNOPSIS
#include <string.h>

char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);

DESCRIPTION
The strcpy() function copies the string pointed to by src, including the terminating null byte ('\0'), to the buffer pointed to by dest. The strings may not overlap, and the destination string dest must be large enough to receive the copy.

The strncpy() function is similar, except that at most n bytes of src are copied. Warning: If there is no null byte among the first n bytes of src, the string placed in dest will not be null-terminated.

If the length of src is less than n, strncpy() writes additional null bytes to dest to ensure that a total of n bytes are written.

RETURN VALUE
The strcpy() and strncpy() functions return a pointer to the destination string dest.
### Appendix II: ASCII

**NAME**

`ascii` - ASCII character set encoded in octal, decimal, and hexadecimal.

**DESCRIPTION**

ASCII is the American Standard Code for Information Interchange. It is a 7-bit code. Many 8-bit codes (such as ISO 8859-1, the Linux default character set) contain ASCII as their lower half. The international counterpart of ASCII is known as ISO 646.

The following table contains the 128 ASCII characters.

<table>
<thead>
<tr>
<th>Oct</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
<td>00</td>
<td>NUL  '\0'</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
<td>01</td>
<td>SOH  (start of heading)</td>
</tr>
<tr>
<td>002</td>
<td>02</td>
<td>02</td>
<td>STX  (start of text)</td>
</tr>
<tr>
<td>003</td>
<td>03</td>
<td>03</td>
<td>ETX  (end of text)</td>
</tr>
<tr>
<td>004</td>
<td>04</td>
<td>04</td>
<td>EOT  (end of transmission)</td>
</tr>
<tr>
<td>005</td>
<td>05</td>
<td>05</td>
<td>ENQ  (enquiry)</td>
</tr>
<tr>
<td>006</td>
<td>06</td>
<td>06</td>
<td>ACK  (acknowledge)</td>
</tr>
<tr>
<td>007</td>
<td>07</td>
<td>07</td>
<td>BEL  '\a' (bell)</td>
</tr>
<tr>
<td>010</td>
<td>08</td>
<td>08</td>
<td>BS   '\b' (backspace)</td>
</tr>
<tr>
<td>011</td>
<td>09</td>
<td>09</td>
<td>HT   '\t' (horizontal tab)</td>
</tr>
<tr>
<td>012</td>
<td>10</td>
<td>10</td>
<td>LF   '\n' (new line)</td>
</tr>
<tr>
<td>013</td>
<td>11</td>
<td>11</td>
<td>VT   '\v' (vertical tab)</td>
</tr>
<tr>
<td>014</td>
<td>12</td>
<td>12</td>
<td>FF   '\f' (form feed)</td>
</tr>
<tr>
<td>015</td>
<td>13</td>
<td>13</td>
<td>CR   '\r' (carriage ret)</td>
</tr>
<tr>
<td>016</td>
<td>14</td>
<td>14</td>
<td>SO   (shift out)</td>
</tr>
<tr>
<td>017</td>
<td>15</td>
<td>15</td>
<td>SI   (shift in)</td>
</tr>
<tr>
<td>020</td>
<td>16</td>
<td>16</td>
<td>DLE  (data link escape)</td>
</tr>
<tr>
<td>021</td>
<td>17</td>
<td>17</td>
<td>DC1  (device control 1)</td>
</tr>
<tr>
<td>022</td>
<td>18</td>
<td>18</td>
<td>DC2  (device control 2)</td>
</tr>
<tr>
<td>023</td>
<td>19</td>
<td>19</td>
<td>DC3  (device control 3)</td>
</tr>
<tr>
<td>024</td>
<td>20</td>
<td>20</td>
<td>DC4  (device control 4)</td>
</tr>
<tr>
<td>025</td>
<td>21</td>
<td>21</td>
<td>NAK  (negative ack.)</td>
</tr>
<tr>
<td>026</td>
<td>22</td>
<td>22</td>
<td>SYN  (synchronous idle)</td>
</tr>
<tr>
<td>027</td>
<td>23</td>
<td>23</td>
<td>ETB  (end of trans. blk)</td>
</tr>
<tr>
<td>030</td>
<td>24</td>
<td>24</td>
<td>CAN  (cancel)</td>
</tr>
<tr>
<td>031</td>
<td>25</td>
<td>25</td>
<td>EM   (end of medium)</td>
</tr>
<tr>
<td>032</td>
<td>26</td>
<td>26</td>
<td>SUB  (substitute)</td>
</tr>
<tr>
<td>033</td>
<td>27</td>
<td>27</td>
<td>ESC  (escape)</td>
</tr>
<tr>
<td>034</td>
<td>28</td>
<td>28</td>
<td>FS   (file separator)</td>
</tr>
<tr>
<td>035</td>
<td>29</td>
<td>29</td>
<td>GS   (group separator)</td>
</tr>
<tr>
<td>036</td>
<td>30</td>
<td>30</td>
<td>RS   (record separator)</td>
</tr>
<tr>
<td>037</td>
<td>31</td>
<td>31</td>
<td>US   (unit separator)</td>
</tr>
<tr>
<td>040</td>
<td>32</td>
<td>32</td>
<td>SPACE</td>
</tr>
<tr>
<td>041</td>
<td>33</td>
<td>33</td>
<td>!</td>
</tr>
<tr>
<td>042</td>
<td>34</td>
<td>34</td>
<td>&quot;</td>
</tr>
<tr>
<td>043</td>
<td>35</td>
<td>35</td>
<td>#</td>
</tr>
<tr>
<td>044</td>
<td>36</td>
<td>36</td>
<td>$</td>
</tr>
<tr>
<td>045</td>
<td>37</td>
<td>37</td>
<td>%</td>
</tr>
<tr>
<td>046</td>
<td>38</td>
<td>38</td>
<td>&amp;</td>
</tr>
<tr>
<td>047</td>
<td>39</td>
<td>39</td>
<td>'</td>
</tr>
<tr>
<td>050</td>
<td>40</td>
<td>40</td>
<td>(</td>
</tr>
<tr>
<td>051</td>
<td>41</td>
<td>41</td>
<td>)</td>
</tr>
<tr>
<td>052</td>
<td>42</td>
<td>42</td>
<td>*</td>
</tr>
<tr>
<td>053</td>
<td>43</td>
<td>43</td>
<td>+</td>
</tr>
<tr>
<td>054</td>
<td>44</td>
<td>44</td>
<td>,</td>
</tr>
<tr>
<td>055</td>
<td>45</td>
<td>45</td>
<td>-</td>
</tr>
<tr>
<td>056</td>
<td>46</td>
<td>46</td>
<td>.</td>
</tr>
<tr>
<td>057</td>
<td>47</td>
<td>47</td>
<td>/</td>
</tr>
<tr>
<td>060</td>
<td>48</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>061</td>
<td>49</td>
<td>49</td>
<td>1</td>
</tr>
<tr>
<td>062</td>
<td>50</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>063</td>
<td>51</td>
<td>51</td>
<td>3</td>
</tr>
<tr>
<td>064</td>
<td>52</td>
<td>52</td>
<td>4</td>
</tr>
<tr>
<td>065</td>
<td>53</td>
<td>53</td>
<td>5</td>
</tr>
<tr>
<td>066</td>
<td>54</td>
<td>54</td>
<td>6</td>
</tr>
<tr>
<td>067</td>
<td>55</td>
<td>55</td>
<td>7</td>
</tr>
<tr>
<td>070</td>
<td>56</td>
<td>56</td>
<td>8</td>
</tr>
<tr>
<td>071</td>
<td>57</td>
<td>57</td>
<td>9</td>
</tr>
<tr>
<td>072</td>
<td>58</td>
<td>58</td>
<td>A</td>
</tr>
<tr>
<td>073</td>
<td>59</td>
<td>59</td>
<td>B</td>
</tr>
<tr>
<td>074</td>
<td>60</td>
<td>60</td>
<td>C</td>
</tr>
<tr>
<td>075</td>
<td>61</td>
<td>61</td>
<td>D</td>
</tr>
<tr>
<td>076</td>
<td>62</td>
<td>62</td>
<td>E</td>
</tr>
<tr>
<td>077</td>
<td>63</td>
<td>63</td>
<td>F</td>
</tr>
<tr>
<td>080</td>
<td>64</td>
<td>64</td>
<td>G</td>
</tr>
<tr>
<td>081</td>
<td>65</td>
<td>65</td>
<td>H</td>
</tr>
<tr>
<td>082</td>
<td>66</td>
<td>66</td>
<td>I</td>
</tr>
<tr>
<td>083</td>
<td>67</td>
<td>67</td>
<td>J</td>
</tr>
<tr>
<td>084</td>
<td>68</td>
<td>68</td>
<td>K</td>
</tr>
<tr>
<td>085</td>
<td>69</td>
<td>69</td>
<td>L</td>
</tr>
<tr>
<td>086</td>
<td>70</td>
<td>70</td>
<td>M</td>
</tr>
<tr>
<td>087</td>
<td>71</td>
<td>71</td>
<td>N</td>
</tr>
<tr>
<td>088</td>
<td>72</td>
<td>72</td>
<td>O</td>
</tr>
<tr>
<td>089</td>
<td>73</td>
<td>73</td>
<td>P</td>
</tr>
<tr>
<td>090</td>
<td>74</td>
<td>74</td>
<td>Q</td>
</tr>
<tr>
<td>091</td>
<td>75</td>
<td>75</td>
<td>R</td>
</tr>
<tr>
<td>092</td>
<td>76</td>
<td>76</td>
<td>S</td>
</tr>
<tr>
<td>093</td>
<td>77</td>
<td>77</td>
<td>T</td>
</tr>
<tr>
<td>094</td>
<td>78</td>
<td>78</td>
<td>U</td>
</tr>
<tr>
<td>095</td>
<td>79</td>
<td>79</td>
<td>V</td>
</tr>
<tr>
<td>096</td>
<td>80</td>
<td>80</td>
<td>W</td>
</tr>
<tr>
<td>097</td>
<td>81</td>
<td>81</td>
<td>X</td>
</tr>
<tr>
<td>100</td>
<td>82</td>
<td>82</td>
<td>Y</td>
</tr>
<tr>
<td>101</td>
<td>83</td>
<td>83</td>
<td>Z</td>
</tr>
<tr>
<td>102</td>
<td>84</td>
<td>84</td>
<td>[</td>
</tr>
<tr>
<td>103</td>
<td>85</td>
<td>85</td>
<td>\</td>
</tr>
<tr>
<td>104</td>
<td>86</td>
<td>86</td>
<td>]</td>
</tr>
<tr>
<td>105</td>
<td>87</td>
<td>87</td>
<td>^</td>
</tr>
<tr>
<td>106</td>
<td>88</td>
<td>88</td>
<td>_</td>
</tr>
<tr>
<td>107</td>
<td>89</td>
<td>89</td>
<td>`</td>
</tr>
<tr>
<td>110</td>
<td>90</td>
<td>90</td>
<td>'</td>
</tr>
</tbody>
</table>

---

ASCII(7)  Linux Programmer’s Manual  ASCII(7)

---
1. Multiple choice questions
A. 2, 3
B. 1, 4
C. 1, 2, 3
D. 3
E. 1, 2

2. C Basics
(a)
```c
void
swap1(char *s1, char *s2) {
    char *tmp = (char *)malloc(len);
    strcpy(tmp, s1);
    strcpy(s1, s2);
    strcpy(s2, tmp);
}
```

(b) Ben's program would have a segmentation fault. This is because
the string buffers are stored in a read-only memory segment and
cannot be overwritten in swap1.

(c)
```c
void
swap2(char **s1, char **s2) {
    char *tmp;
    tmp = *s1;
    *s1 = *s2;
    *s2 = tmp;
}
```

3. Assembly
(a) mystery function takes two arguments. We know this because mystery function uses
%rsi and %rdi without initialization (and %rsi and %rdi contain argument values by
the C calling convention) The first argument has type node *, second argument has type long.
(b) mystery function has one return value and the type is node *.
(c)
```c
node *
mystery(node *head, long v) {
    node *n = head;
    while (n!= NULL) {
        if (n->value == v) {
            break;
        }
        n = n->next;
    }
    return n;
}
```

4. C and Buffer Overflow
(a)
```c
int
replace(char *str, char *s1, char *s2) {
    char *s = strstr(str, s1);
    if (!s) return 0;
    char *tmp = (char *)malloc(strlen(str) + strlen(s2));
    strcpy(tmp, str);
    char *tmps = strstr(tmp, s1);
    strcpy(tmps, s2);
    strcpy(tmps+strlen(s2), s+strlen(s1));
    strcpy(str, tmp);
    free(tmp);
    return 1;
}
```

(b) "hello x" with a terminating null character
(c) 0x4007fc "x"
0x4007f4 "friends and families"
(d) 40075e
(e) 2.