CS/ECE 354 - Practice Midterm Exam Spring 2016

C Programming

1. The reason for using pointers in a C program is

    a. Pointers allow different functions to share and modify their local variables.
    b. To pass large structures so that complete copy of the structure can be avoided.
    c. Pointers enable complex “linked" data structures like linked lists and binary trees.
    d. All of the above.

2. Assume that an int variable takes 4 bytes and a char variable takes 1 byte. What is the output of the code below?

```
#include <stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr1 = arr;
    int *ptr2 = arr + 5;
    printf("Number of elements between two pointer are: %d.",
                                                (ptr2 - ptr1));
    printf("Number of bytes between two pointers are: %d",
                              (char*)ptr2 - (char*)ptr1);
    return 0;
}
```

    a. Number of elements between two pointer are: 5. Number of bytes between two pointers are: 20
    b. Number of elements between two pointer are: 20. Number of bytes between two pointers are: 20
    c. Number of elements between two pointer are: 5. Number of bytes between two pointers are: 5
    d. Compiler Error

e. Runtime Error

3. What would be the output of following code. Assume size of an int on this machine is 4 bytes and size of a char is 1 byte. (Alert: It will not be a compiler error!).

```
int main (void)
{
  char arr[] = "MachineOrganization";

  int * ptr = (int *) (&arr[1]);
  printf ("value is %c\n",  *((char *)ptr));
  ptr++;
  printf ("value is %c\n",  *((char *)ptr));
  return 0;
}
```

Output:
value is _____
value is _____

4. How does the following function modify the linked list shown below?

LINKED LIST: head → 1 → 2 → 3 → 4 → 5 → NULL

```
void ultimate_fun(struct node **head_ref)
{
    if (*head_ref == NULL || (*head_ref)->next == NULL)
        return;

    struct node *secLast = NULL;
    struct node *last = *head_ref;

    while (last->next != NULL)
    {
        secLast = last;
        last = last->next;
    }

    secLast->next = NULL;
```

```
        last->next = *head_ref;

        *head_ref = last;
    }
```

a. head → 1 → 2 → 3 → 4 → 5 → NULL
b. head → 5 → 1 → 2 → 3 → 4 → NULL
c. head → 2 → 3 → 4 → 5 → 1 → NULL
d. head → 5 → 2 → 3 → 4 → 1 → NULL

5. What is the value of n for the linked list shown below after the function do_something( ) is called with the arguments as shown below?

```
LINKED LIST: head -> 10 -> 20 -> 30 -> 40 -> 50 -> NULL

int n = do_something(head, 30);

 int do_something(struct node *head, int element)
{
    struct node *temp = head;
    int pos = 0;
    if (head == NULL) {
        return -1;
    } else {
        while (temp != NULL) {
            ++pos;
            if (temp->data == element) {
                return pos;
            }
            temp = temp->next;
        }
        return -1;
    }
}
```

e. n = 3
f. n = 30
g. n = 2
h. n = -1

Data Representation

6. The following is a memory layout of an integer number on a new machine developed at Wisconsin Madison.

Memory Location	Value
0x200	AB
0x201	F3
0x202	17
0x203	9E

What is the value of this integer in hexadecimal, if this machine is **little** endian?

What is the value of this integer in hexadecimal, if this machine is **big** endian?

7. What is the issue with the code below and how will you fix it?

```
#define BUFF_SIZE 10
int main(int argc, char* argv[]){
      int len;
      char buf[BUFF_SIZE];
      len = atoi(argv[1]);
      if (len < BUFF_SIZE){
            memcpy(buf, argv[2], len);
      }
}
```

The following information about atoi and memcpy is given:

```
int atoi(const char *nptr);
void *memcpy(void *dest, const void *src, unsigned int n);
```

Assembly Programming

Note: 3 more practice questions in Assembly were already discussed in class on Friday, March 11th 2016!

8. Assembly to C

   char someFunc(int a, int b, int c);

   The arguments for this function are on the stack at the following addresses:

   int a ⇒ `0x8(%ebp)`
   int b ⇒ `0xc(%ebp)`
   int c ⇒ `0x10(%ebp)`

   Complete the following line with a valid C expression.

   char t5 = _____

```
 4e: 8b 45 0c              mov    0xc(%ebp),%eax
51: 8b 55 08               mov    0x8(%ebp),%edx
54: 89 d1                  mov    %edx,%ecx
56: 29 c1                  sub    %eax,%ecx
58: 89 c8                  mov    %ecx,%eax
5a: 85 c0                  test   %eax,%eax
5c: 0f 9f c0               setg   %al
5f: 88 45 fe               mov    %al,-0x2(%ebp)
```

9. Consider the following assembly code for a C function named loops( ). The line numbers are given in decimal.

```
char loops(int a, int b);

1:          push   %ebp
2:          mov    %esp,%ebp
3:          sub    $0x10,%esp

            //int sumA = a;
4:          mov    0x8(%ebp),%eax
```

```
5:          mov    %eax,-0x8(%ebp)

            //int sumB = b;
6:          mov    0xc(%ebp),%eax
7:          mov    %eax,-0x4(%ebp)

8:          jmp    .L3
      .L1:
9:          mov    -0x8(%ebp),%eax
10:         cmp    -0x4(%ebp),%eax
11:         jge    .L2

12:         mov    0x8(%ebp),%eax
13:         add    %eax,-0x8(%ebp)
14:         jmp    .L3

      .L2
15:         mov    0xc(%ebp),%eax
16:         add    %eax,-0x4(%ebp)

      .L3:
17:         mov    -0x8(%ebp),%eax
18:         cmp    -0x4(%ebp),%eax
19:         jne    .L1

20:         mov    -0x8(%ebp),%eax
21:         leave
22:         ret
```

Complete its corresponding C function (based on the above assembly program) by filling in the blank lines.

```
char loops(int a, int b)
{
  int sumA = a;
  int sumB = b;
  while(_______)
  {
    if(__________)
```

```
      __________
    else
      __________
  }
  return sumA;
}
```

10. Write the C code for the following assembly language code.

    Note: The C code can be written using less than 5 lines of code.

```
mystery:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    8(%ebp), %edx
    movl    12(%ebp), %eax
    addl    %edx, %eax
    movl    %eax, -4(%ebp)
    movl    8(%ebp), %eax
    subl    12(%ebp), %eax
    movl    %eax, -8(%ebp)
    movl    -4(%ebp), %eax
    imull   -8(%ebp), %eax
    movl    %eax, -12(%ebp)
    movl    -12(%ebp), %eax
    leave
    ret

int mystery(int x, int y)
{




}
```