Module 4
C, Objective C and SWIFT - MADP 401 (140 hours/7 weeks)

Assignment 8 (in-class activity)

Due: 5:30pm, Feb 1st, 2017

Requirements

- N/A

Problem1

- Write a program, which allows the user to enter some positive numbers (numbers which are greater than 0). Once the user enters 0 or a negative number it would stop accepting inputs from the user. Then using dynamic memory allocation, find the biggest number entered by the user and print it.

Problem2

- Transform the following piece of code to use only array notation (and index manipulation) so that there is no * and & notation in the code anymore.

```
void main (void)
{
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int *ptr = &arr[0];
    while (ptr <= &arr[9])
    {
        printf("ptr points to %d\n", *ptr);
        ptr++; // ptr now points to the next element
    }

}
```

Problem3

- Understanding memory leak:
- Look at the following snippet of code:

```
int n=10;
int a[n], i=0;
for(int i=0; i<n; i++)
{
    a[i] = i+1;
}
int* b = (int*) malloc (2*n*sizeof(int));
b = a;
```

- Explain how memory leak happens as the result of running this code.

Problem4

- Fixed and dynamic collection of strings:
  - A) Read 5 lines of arbitrary length from input. The end of a line is until the user enters \n: Use array and fixed memory size for this.
  - B) Read up to 5 lines of arbitrary length from input. The end of a line is until the user enters \n. Solve this problem using malloc but without using realloc.
  - C) Read unlimited number of lines of arbitrary length from input. The end of a line is until the user enters \n. When user enters \n on an empty line the last non-empty line will be the last one.
  - Write a function called fixedLines, which returns the length of the longest entered line in part A.
  - Write a function called semiFixedLines, which returns the length of the longest entered line in part B.
  - Write a function called dynamicLines, which returns the length of the longest entered line in part C.

Problem5

- Multiple Choice Questions: For each question, draw the debugging the table and memory (if needed) on a piece of paper and find the answer. Then write the code and run it and verify your answer. The TA will check your debugging table and memory as well as the code.

- 5.1 What is the output of this C code?

```
1.  #include <stdio.h>
2.  void main()
3.  {
4.      m();
5.      m();
6.  }
7.  void m()
8.  {
9.      static int x = 5;
10.     x++;
11.     printf("%d", x);
12. }
```

a) 6 7

b) 6 6

c) 5 5

d) 5 6

5.2- What is the output of this C code?

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      foo();
5.      foo();
6.  }
7.  void foo()
8.  {
9.      int i = 11;
10.     printf("%d ", i);
11.     static int j = 12;
12.     j = j + 1;
13.     printf("%d\n", j);
14.  }
```

a) 11 12 11 12

b) 11 13 11 14

c) 11 12 11 13

d) Compile time error

5.3- Comment on the output of this C code?

```
1.  #include <stdio.h>
```

```
2.      void func();
3.      int main()
4.      {
5.          static int b = 20;
6.          func();
7.      }
8.      void func()
9.      {
10.         static int b;
11.         printf("%d", b);
12.     }
```

a) Output will be 0

b) Output will be 20

c) Output will be a garbage value

d) Compile time error due to redeclaration of static variable

5.4- What is the output of this C code?

```
1.      #include <stdio.h>
2.      #define foo(m, n) m ## n
3.      int main()
4.      {
5.          printf("%s\n", foo(k, l));
6.      }
```

a) k l

b) kl

c) Compile time error

d) Undefined behaviour

5.5- What is the output of this C code?

```
1.      #include <stdio.h>
2.      #define foo(x, y) #x #y
3.      int main()
4.      {
5.          printf("%s\n", foo(k, l));
6.          return 0;
7.      }
```

a) kl

b) k l

c) xy

d) Compile time error

5.6- What is the output of this C code?

```
1.  #include <stdio.h>
2.  void main()
3.  {
4.      #define max 37
5.      printf("%d", max);
6.  }
```

a) 37

b) Run time error

c) Varies

d) Depends on compiler

5.7- What is the output of this C code?

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      double *ptr = (double *)100;
5.      ptr = ptr + 2;
6.      printf("%u", ptr);
7.  }
```

a) 102

b) 104

c) 108

d) 116

5.8- Comment on the output of this C code?

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int *p = (int *)2;
5.      int *q = (int *)3;
```

```
6.          printf("%d", p + q);
7.      }
```

a) 2

b) 3

c) 5

d) Compile time error

5.9 What is the output of this C code?

```
1.      #include <stdio.h>
2.      void main()
3.      {
4.          char *s = "hello";
5.          char *p = s;
6.          printf("%c\t%c", *(p + 1), s[1]);
7.      }
```

a) h e

b) e l

c) h h

d) e e

5.10 What is the output of this C code (considering sizeof char is 1 and pointer is 4)?

```
1.      #include <stdio.h>
2.      int main()
3.      {
4.          char *a[2] = {"hello", "hi"};
5.          printf("%d", sizeof(a));
6.          return 0;
7.      }
```

a) 9

b) 4

c) 8

d) 10

5.11 What is the output of this C code?

```
1.      #include <stdio.h>
2.      int main()
```

```
3.      {
4.          char a[2][6] = {"hello", "hi"};
5.          printf("%s", *a + 1);
6.          return 0;
7.      }
```

a) hello

b) hi

c) ello

d) ello hi

5.12- What is the output of this C code?

```
1.      #include <stdio.h>
2.      void main()
3.      {
4.          char *p = calloc(100, 1);
5.          p = "welcome";
6.          printf("%s\n", p);
7.      }
```

a) Segmentation fault

b) Garbage

c) Error

d) welcome

5.13 For the following program, Which of the following should be used for freeing the memory allocated?

```
1.      #include <stdio.h>
2.      struct p
3.      {
4.          struct p *next;
5.          int x;
6.      };
7.      int main()
8.      {
9.          struct p *p1 = (struct p*)malloc(sizeof(struct p));
10.         p1->x = 1;
11.         p1->next = (struct p*)malloc(sizeof(struct p));
12.         return 0;
```

```
13.     }
```

a) free(p1);
   free(p1->next)
b) free(p1->next);
   free(p1);
c) free(p1);
d) All of the mentioned

5.14- What is the output of this C code?

```
1.    #include <stdio.h>
2.    struct p
3.    {
4.        struct p *next;
5.        int x;
6.    };
7.    int main()
8.    {
9.        struct p *p1 = calloc(1, sizeof(struct p));
10.       p1->x = 1;
11.       p1->next = calloc(1, sizeof(struct p));
12.       printf("%d\n", p1->next->x);
13.       return 0;
14.   }
```

a) Compile time error
b) 1
c) Some garbage value
d) 0

5.15- What is the output of this C code?

```
1.    #include <stdio.h>
2.    struct p
3.    {
4.        struct p *next;
5.        int x;
6.    };
7.    int main()
8.    {
```

```
9.          struct p* p1 = malloc(sizeof(struct p));
10.          p1->x = 1;
11.          p1->next = malloc(sizeof(struct p));
12.          printf("%d\n", p1->next->x);
13.          return 0;
14.      }
```

a) Compile time error

b) 1

c) Some garbage value

d) 0

Bonus Problem (optional)

Remember the bonus problem for previous assignment (assignment 6)? Solve the same problem using dynamic memory allocation.