PASS MOCK EXAM

– FOR PRACTICE ONLY –

Course: SYSC 2006 **Facilitator: Lilly M**

Dates and locations of final exam take-up:
Friday Dec. 4th (6:00-9:00pm) TB 240

It is **most beneficial** to you to write this mock final **UNDER EXAM CONDITIONS**. This means:

1 • Complete the final in _3_ hour(s).
2 • Work on your own.
3 • Keep your notes and textbook closed.
4 • Attempt every question.

After the time limit, go back over your work with a different colour or on a separate piece of paper and try to do the questions you are unsure of. Record your ideas in the margins to remind yourself of what you were thinking when you take it up at PASS.

The purpose of this mock exam is to give you practice answering questions in a timed setting and to help you to gauge which aspects of the course content you know well and which are in need of further development and review. Use this mock exam as a ***learning tool*** in preparing for the actual exam.

Please note:

- Come to the PASS session with your mock exam complete. There, you can work with other students to review your work.
- Often, there is not enough time to review the entire exam in the PASS session. Decide which questions you most want to review – the Facilitator may ask students to vote on which questions they want to discuss.
- Facilitators do not bring copies of the mock exam to the session. Please print out and complete the exam before you attend.
- **Facilitators do not produce or distribute an answer key for mock exams.** Facilitators help students to work together to compare and assess the answers they have. If you are not able to attend the PASS session, you can work alone or with others in the class.

Good Luck writing the Mock Exam!!

1) Arrays [15 marks]

a) Your supervisor has asked you to help create a software game for pre-schoolers trying to learn compound words. Your task is to write the code segment that determines whether a given word contains the word 'and' . Write a function that takes in the parameters: *word*, which is a char array of n-elements, and *n* the number of elements (characters) contained in the array word. The function should return 'true' if the function contains the word 'and' and 'false' if the function does not contain the word 'and'. Use pointers to achieve your task. Assume all words are lower case. [7 marks]

_Bool and (char word [], int n);

Ex. The if an array containing the following words is passed to 'and' the result should be as follows:

command function returns 'true'
cannon function returns 'false'
kindergarten function returns 'false'

b) After approving your above code, your boss now asks you to write a function to show grade two students how to arrange the numbers in order from zero to ten. If the students are given a list of numbers which contains a number greater than 10 they are to replace it with the number ten and put in in order in the list. If a number is negative, it's absolute value is taken and it is placed in order in the array. Your function is passed the following parameters: 'list' which is an array of integer numbers, and n, which is the number of elements of the passed array 'list'. The function is to return an array with the numbers in order from zero to ten and print this on the screen. [8 marks]

Ex. int* zerototen (int list [], int n)

If the function is passed the following arrays for 'list' the output should be as follows:

If passed [-1, 10, 12, 6, 13, 18] the function returns [1, 6,10, 10, 10, 10]
If passed [-1, 10, 12, 6, -13, 0, 18] the function returns [0, 1, 6,10, 10, 10, 10]
If passed [-1, -1, 12, 6, -13, 0, 5,5] the function returns [0, 1, 1, 5,5 6,10, 10]

2) Recursion [10 marks]

The following function determines the number of bacteriophages present in a given 10 minute interval time period given by n:

$$B=2n$$

a) You are required to write a function which is passed the 'n' value and the a_1 value and returns the number of bacteriophages present after the time interval. [5 marks]

int bacteriophage (int a_1, int n);

Ex. If you were given the value of $a_1=2$ the following sequence would be produced for n=3: 2,4,6

b) Draw the recursive memory diagram if you were given the $a_1=6$ and n=3. [5 marks]

3) Linked Lists [15 marks]

Assume the conventions and methods discussed in class and provided on the crib sheet.

a) You are given an array which contains integer values. Convert this array into a singly linked-list, create a new linked list that is initialized with the elements in the array of integers (one integer per node). The numbers in the linked list appearing in the same order that they appear in the array. [6 marks]

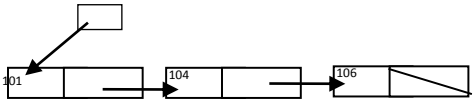
b) Create a function which is passed a singly linked-list named 'list' and returns a pointer to linked-list with the values in 'list' in order from least to greatest. Hint: you must copy the linked list to a malloc'd array, sort the array, deallocate the original linked list, and call the function from (a) to create a new linked list. [9 marks]

4) Queues [8 marks]

Customers often have to wait for several minutes before they can talk to customer service telephone operator, as they must wait in a queue of callers until their turn. Taking this into account Bell Media has created a new policy which assigns each customer a number and prints a message in their system which states how many people are waiting to be served, who is currently being served and who will be served next. Write a function named 'wait_time' which is passed a singly-linked list named 'list' and 'n' which is the number of customers in the queue. Remove a customer after they are served and ensure your code segment only stops when there are no more people in the queue. You must use linked-lists to achieve your task. Assume linked-list is composed of nodes which follow the 'Intnode' structure defined in class.

void wait_time(intlist * list, int n);

Ex. If you are passed the following linked list



The function would output the following:

"There are currently 3 people waiting to be served."
"Person number 101 is now being served and person number 104 will be served next."
"There are currently 2 people waiting to be served."
"Person number 104 is now being served and person 106 will be served next."
"There is currently 1 person waiting to be served."
"Person number 106 is now being served, there are no other customers at this time."

Memory Diagrams [10 marks]:

Trace through the following code segment, stopping right after each labelled point

```
#include <stdio.h>
int math_one(int link, int *new)
{
        int prod;
        prod =link*2 – *new ;
        link = (*new) *12+1 –prod;
        *new = prod++ * link; /* Point A */
        return prod; /*Point B*/
}

void math_three (int p[], int n, int *there, int their)
{
        int theyre=1;
        p[n-1* 2] = 1;
        *there = p[n-4-1] - p[1];
        their = *there / (theyre*6);
        /* Point C */
        return;
}

int main(void)
{
        int list[6];
        int j, m, n;
        list[0] = 8; list[1] = -1; list[2] = 0; list[3] = 12; list [5]=100; list[4]=12;
        m = 12;
        n = 1;

        j = do_this(m, &n);/* Point C */
        m = j;
        n = 1;
        do_that(list, 6, &m, n); /* Point D */
      return 0;
}
```

A) Draw the corresponding memory diagram up to and including whatever has been executed at the semi-colon that is immediately before /*Point A* / .

B) Draw the corresponding memory diagram up to and including whatever has been executed at the semi-colon that is immediately before /*Point B* / .

C) Draw the corresponding memory diagram up to and including whatever has been executed at the semi-colon that is immediately before /*Point C* / .

D) Draw the corresponding memory diagram up to and including whatever has been executed at the semi-colon that is immediately before /*Point D* / .

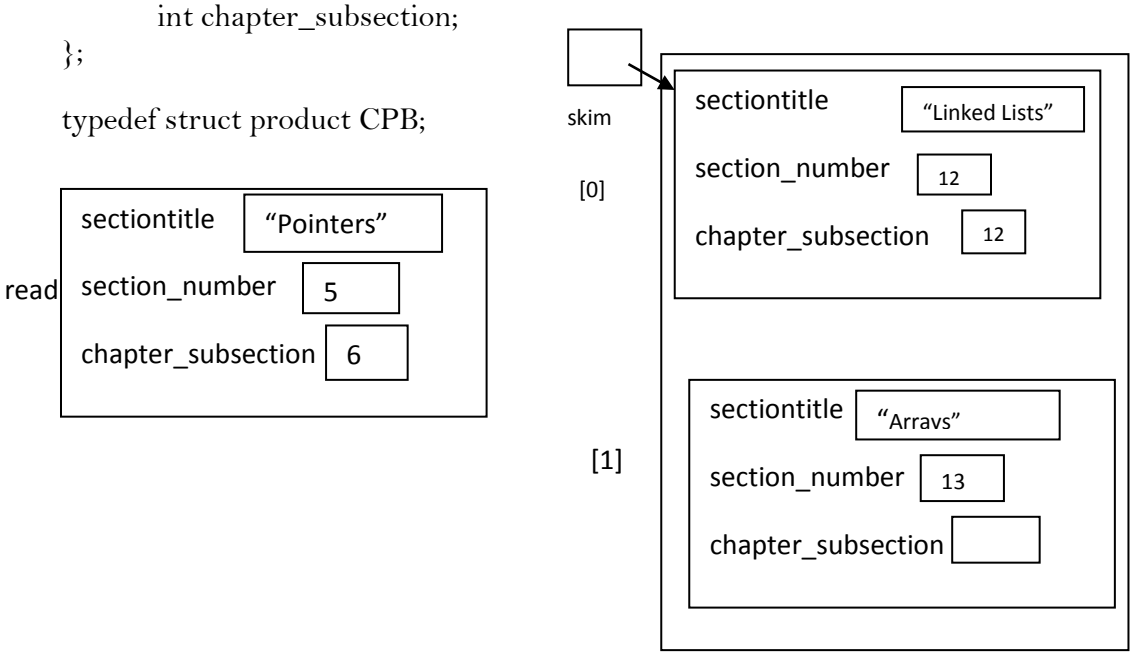
Memory Diagrams with Structures

Here is the declaration for a structure which describes a netbook's files:

```
struct ComputerProgrammingforBeginners {

        char*sectiontitle;
        int section+number
        int chapter_subsection;
};

typedef struct product CPB;
```



Write the code that when executed produces the memory diagram above.

a) Write the C code to dynamically allocate an array from the heap which can store the chapter information of the two chapters above (i.e. in [0] and [1]). Do not write a complete function

b) Write the C code necessary to initialize the values in 'read' as shown. With information about the pointers chapter.

c) Modify the memory diagram on the page so it shows what it would look like after the following code segment is executed:

```
CPB* n;
n=&skim[0];
*n=read;
```

d) Initalize chapter_subsection in skim[1] to 100.