RECURSION EXERCISES

Tracing and Programming Problems

PRELIMINARY NOTES. I have organized this exercise set as an alternating sequence of TRACING PROBLEM, followed by a PROGRAMMING PROBLEM.

- The primary objective of the Tracing Problems is to show you first some examples of recursive functions. MANUALLY tracing the sample function will help you learn how to READ, ANALYZE and UNDERSTAND important programming concepts and techniques.
- Questions are formulated to help guide you in analyzing and understanding the codes.
- The programming problems are then provided to reinforce and let you apply the concepts and techniques that you learned from doing the tracing problems.

PROBLEM #1-A: Tracing Problem.

```
int Mystery(int nNum)
{
      if (nNum < 10)
         return 1;
      else
         return Mystery(nNum/10) + 1;
}
```

1. What is the base case condition?
2. What is the recursive case condition?
3. Is the function tail-end or non-tail end recursive?
4. What is the value of Mystery(58)? Mystery(724)? Mystery(904375)?
5. What is the value of nNum/10 when nNum is 58? when nNum is 724? when nNum is 904375?
6. What is the use or purpose of the expression nNum/10 in the recursion above?
7. What does Mystery() do? Note: Don't simply say it returns a value. You need to explain what kind of value or what is the nature of the value that the function returns.
8. What do you think is the meaning or use of "**1**" which is the value returned in the base case?
9. What do you think is the meaning or use of the "**+ 1**" in the recursive case?
10. Verify your answers to question 4 and 7 by encoding and runnning the function. Write a main() function that will call the Mystery function(). An example main() function is shown below for your reference.

```
int main()
{
    int nNum;

    printf("Input a positive integer: ")
    scanf("%d", &nNum);
    printf("Mystery(%d) is %d.\n", nNum, Mystery(nNum));
    return 0;
}
```

PROBLEM #1-B: Programming Problem.

Open the file named "**prob_1B.c**". Implement the required function based on what you learned from the previous tracing problem. Make sure that you test your solution thoroughly using different input values.

PROBLEM #2-A: Tracing Problem.

```
void Mystery(int nNum)
{
    if (nNum < 10) {
       if (nNum % 2)
          printf("%d", nNum);
    }
    else {
       Mystery(nNum/10);

       if ( (nNum % 10) % 2 )
          printf("%d", nNum % 10);
   }
}
```

1. What is the base case condition?
2. What is the recursive case condition?
3. Is the function tail-end or non-tail end recursive?
4. What is the output corresponding to Mystery(492)? Mystery(15236)? Mystery(98765431)?
5. What is use or purpose of "**if ((nNum % 10) % 2)**" in the recursive case?
6. What kind of digits are printed by Mystery()?
7. Verify your answers to question 4 and 6 above by encoding and runnning the function above Write a main() function that will call the Mystery function().

Next, edit the function above such that the sequence of the statements in the else clause is interchanged. That is, the else clause will now be:

```
else {
     if ( (nNum % 10) % 2 )
         printf("%d", nNum % 10);

     Mystery(nNum/10);
}
```

8. Is the modified function tail-end or non-tail end recursive?
9. What is the output corresponding to Mystery(15236)?
10. What is the output corresponding to Mystery(987654321)?
11. What do you think is the basic difference between the original function and the modified function?

PROBLEM #2-B: ProgrammingProblem.

Open the file named "**prob_2B.c**". Implement the required function based on what you learned from the previous tracing problem. Make sure that you test your solution thoroughly using different input values.

PROBLEM #3-A: Tracing Problem.

```
int Mystery(int nNum, int temp)
{
    if (nNum < 10) {
       if (nNum < temp)
          temp = nNum;

       return temp;
    }
    else {
       if (nNum % 10 < temp)
          temp = nNum % 10;

       return Mystery(nNum/10, temp);
   }
}
```

1. What is the base case condition?
2. What is the recursive case condition?
3. Is the function tail-end or non-tail end recursive?
4. What is the value of Mystery(482, 9)? Mystery(51236, 9)? Mystery(982763, 9)?
5. What do you think is the reason why the first call to Mystery(nNum, temp) has a value of 9 for its second parameter (temp)?
6. In successive recursive calls, what condition/s will result into a change in the value of temp?
7. What does Mystery() do?
8. Verify your answers to question 4 and 7 above by encoding and runnning the function above Write a main() function that will call the Mystery function(). DON'T forget to call Mystery() with two parameters. The second parameter in your function call should initially be 9 for the function to work correctly.

PROBLEM #3-B: Programming Problem.

Open the file named "**prob_3B.c**". Implement the required function based on what you learned from the previous tracing problem. Make sure that you test your solution thoroughly using different input values.

PROBLEM #4-A: Tracing Problem.

```
void Mystery(int nNum, int nPlace)
{
     if (nNum < 10)
        printf("%d %ds\n", nNum, nPlace);
     else {
        printf("%d %ds\n", nNum % 10, nPlace);
        Mystery(nNum/10, nPlace * 10);
     }
}
```

1. What is the base case condition?
2. What is the recursive case condition?
3. Is the function tail-end or non-tail end recursive?
4. What is the output corresponding to Mystery(5, 1)? Mystery(23, 1)? Mystery(8064, 1)? Mystery(432109, 1)?
5. What do you think is the reason why the first call to Mystery(nNum, nPlace has a value of 1 for its second parameter (nPlace)?
6. What does Mystery() do?
7. Verify your answers to questions 4, 5 and 6 above by encoding and runnning the function above Write a main() function that will call the Mystery function(). Make sure that you have two parameters, with a value of 1 for the second parameter.

PROBLEM #4-B: Programming Problem.

Open the file named "**prob_4B.c**". Implement the required function based on what you learned from the previous tracing problem. Make sure that you test your solution thoroughly using different input values.

Ending Questions.

1. Is there a need for a return statement for recursive functions that are of type void?
2. Is there a need for a return statement for recursive functions that are of type non-void?
3. Is there a need for a return statement in the base case of functions that are of type non-void?
4. Is there a need for a return statement in the recursive case for functions that are of type non-void?

– end of this document –