Surprise Test 1 Solutions (Sec B) Max: 2 Marks. Time: 20 Minutes

Name: _____ Roll Number: _____

Note: Negative mark of -1 applies for each incorrect or incomplete response.

Question 1. I was writing a program to to play the 21 matchstick game. But, a mischievous friend of mine deleted one line. Can you fill in the blanks to complete this code and make it work? [1 Mark]

```
int main()
{
    int stickCount=21, x, y;

    while (stickCount>=1)
    {
        printf("\nNumber of sticks left = %d. Your turn now.
                    Pick up 1-4 matchsticks: ", stickCount);
        scanf ("%d", &x);

        if (x>4)
        {
            printf("Sorry. Select between 1 and 4.\n");
            break;
        }

        y = 5-x;

        if (y > stickCount) {
            y = stickCount - 1; //Write your answer here.
        }

        printf("System chooses: %d", y;
        stickCount=stickCount-(x+y);
        if(stickCount==1) break;
    }

    printf("\nOnly one stick remains. You must select it. So, system Wins.");
}
```

Answer: You may write anything that compiles!

Explanation: stickCount variable takes the number of match sticks still left to be picked up. While there is at least one stick left (which will always be the case), we let user pick first. x denotes the number of sticks user picks in every iteration. User is not allowed to pick more than 4. The code that checks this is called input validation. It is always good to validate input. This code validates only the upper bound. It is also good to validate the lower bound. y denotes the number of sticks the system should pick. Since the sum should always be five, system picks y = 5 - x sticks. So, we start with 21 sticks, come to 16 after user and system picks sticks in the first round. 16 becomes 11 after second round. 11 becomes 6. If user picks x=2 sticks now, we have y = 5 - 2 = 3 sticks. In any case, it can be proved that y will never take a value greater than stickCount. Hence the code inside the if condition comparing y with stickCount will never be satisfied. So, the code inside will never be executed. We call such code that never gets executed as dead code. Hence, you may write anything here and it will work just fine as long as there are no compilation issues!

Question 2. What is the output? [1 Mark]

```
int fun(int n, int *y)
{
   int t, f;
   if(n <= 1)
   {
     *y = 1;
      return 1;
   }
   t = fun(n-1, y);
   f = t + *y;
   *y = t;
   return f;
}
int main( )
{
  int x = 15;
  printf ( "%d\n", fun (6, &x));
  getchar();
  return 0;
}
```

Answer = 13

Explanation: This problem combines the idea of recursion along with pointers as arguments to the function. You may create a project in codeblocks and debug this line by line to get clarity on how this works.