3) (10 pts) DSN (Bitwise Operators)

A company is looking to hire an employee based on answers to a questionnaire. The questionnaire has 20 yes/no questions (labeled from question 0 to question 19) and an applicant's set of responses is stored as a single integer such that the i^{th} bit is set to 1 if the response to question i is yes, and it's set to 0 if the response to question i is no. For example, if an applicant answered yes to questions 0, 2, 3, 5, 8, and 10 and no to the other 14 questions, her responses would be stored as the single integer 1325, since 00000000010100101101_2 = 1325. For all questions, the company prefers yes answers to no answers. However, since it's unlikely that a candidate will answer yes to all of the questions, they have developed a modified scoring system for each candidate. The company has ranked each of the 20 questions in order of preference from most important to least important. This ranking is an array, ***preferences***, that stores a permutation of the integers from 0 to 19, inclusive. For example, if preferences[0] = 8, preferences[1] = 10 and preferences[2] = 1, then the company cares most about the answer to question 8, second most about the answer to question 10 and third most about the answer to question 1. A candidate's score is simply the maximum number of the most important questions they answered yes without a single no. For example, the sample candidate described above would have a score of 2, because she said yes to questions 8 and 10, but no to question 1, which was third on the preference list. Any candidate who said no to question 8 would be assigned a score of 0. Complete the function below so it returns the score of the applicant whose answers are stored in the formal parameter ***applicant***. The formal parameter ***preferences*** is an array of size 20 which stores the order of importance of the questions as previously described. **(Note: You must use bitwise operators to earn full credit for this question.)**

```
#define SIZE 20

int score(int preferences[], int applicant) {


    int res = 0;                          // Grading: 1 pt
    for (res=0; res<SIZE; res++)          // Grading: 2 pts
        if ( (applicant & (1<<preferences[res])) == 0) // 5 pts
            break;                        // Grading: 1 pt

    return res;                           // Grading: 1 pt

}
```

Grading Note: Student answers may look different than this. 4 pts are awarded for the general set up of having an variable keeping track of the answer, returning it and loop through the search space. 6 pts are allocated for the logic of deciding when to stop. We can break down those points as follows:

2 pts for accessing the right question preference
1 pt for bit shifting this value or equivalent (could right shift applicant…)
2 pts for taking the former and bitwise anding it with applicant
1 pt for stopping the loop at the appropriate time