1) (10 pts) DSN (Dynamic Memory Management in C)

This problem relies on the following struct definition:

```
typedef struct Employee
{
  char *first; // Employee's first name.
  char *last;  // Employee's last name.
  int ID;      // Employee ID.
} Employee;
```

Consider the following function, which takes three arrays – each of length *n* – containing the first names, last names, and ID numbers of *n* employees for some company. The function dynamically allocates an array of *n* Employee structs, copies the information from the array arguments into the corresponding array of structs, and returns the dynamically allocated array.

```
Employee *makeArray(char **firstNames, char **lastNames, int *IDs, int n)
{
  int i;
  Employee *array = malloc(__________________________________________);

  for (i = 0; i < n; i++)
  {
    array[i].first = malloc(________________________________________________);

    array[i].last = malloc(_________________________________________________);

    strcpy(array[i].first, firstNames[i]);
    strcpy(array[i].last, lastNames[i]);
    array[i].ID = IDs[i];
  }

  return array;
}
```

a) Fill in the blanks above with the appropriate arguments for each *malloc()* statement.

b) Next, write a function that takes a pointer to the array created by the *makeArray()* function, along with the number of employee records in that array (*n*) and frees all the dynamically allocated memory associated with that array. The function signature is as follows:

```
void freeEmployeeArray(Employee *array, int n)
{
```