

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

The struct, dataTOD, shown below, is used to collect data from different devices connected to the CPU. Every time the data is updated a new buffer containing the structure's data is created and populated.

```
typedef struct dataTOD{
    int seconds;        // seconds since midnight
    double data;        // data sample
    char * dataName;    // data name (optional)
} dataTOD;
```

- (a) (8 pts) Write the code necessary to create and initialize the members of dataTOD in a function named `init_dataTOD` that returns a pointer to the newly created buffer. Return a NULL in the event a buffer cannot be created. Otherwise, set the seconds and data values according to the corresponding input parameters to `init_dataTOD`, dynamically allocate the proper space for `dataName` and then copy the contents of name into it (not a pointer copy) and return a pointer to the newly created struct.

```
dataTOD * init_dataTOD(int sec, double val, char* name){

    dataTOD * tDta = malloc(sizeof(dataTOD));
    if (tDta == NULL)
        return NULL;
    tDta->seconds = sec;
    tDta->data = val;
    tDta->dataName = malloc((strlen(name)+1)*sizeof(char));
    strcpy(tDta->dataName, name);
    return tDta;
}
```

Grading: 1 point for each line shown above. Assign partial if necessary but assign a whole number of points.

- (b) (2 pts) Complete the function below so that it frees all the dynamically allocated memory pointed to by its formal parameter `zapThis`. You may assume that the pointer itself is pointing to a valid struct and its `dataName` pointer is pointing to a dynamically allocated character array.

```
void free_dataTOD(dataTOD *zapThis){
    free(zapThis->dataName);
    free(zapThis);
}
```

Grading: 1 pt each line