COMP 132: Advanced Programming
Homework 8

Dynamic memory allocation in C and Programming in large

These homework questions focus on self-referential structures, dynamic memory allocation functions of C and programming in large.

Create a user-defined library for the following user-defined type definitions, constants, and the required function prototypes in the header file (worker.h). Put all the function implementations in the source code file (worker.c). Put the main function in another source code file (test.c).

1. Define an enum (enumeration constants) named DEPT representing the department name codes as follows.

```
enum deptcode { sales, finance, packing, engineering };

typedef enum deptcode DEPT;
```

2. Define a global static array of strings to keep the department names as follows. You can use this array to print the string equivalence of each enum defined above.

```
static char *deptNames[] = { "Sales","Finance","Packing","Engineering" };
```

3. Define a user-defined type (struct) named **Worker** with following fields:

int id: represents the unique identifier of the Worker

char name[30]: represents the name of the Worker

int age: represents the age of the Worker

DEPT department: represents the department code where the Worker is registered

int salary: represents the salary of the Worker

struct Worker *next: represents the next Worker in the list of Workers

4. Define a function with the following prototype:

```
void addWorker(ListPtr *listPtr, int id, char *name, int age, DEPT
department, int salary);
```

which adds a Worker with the specified data (id, name, age, department and salary given in the arguments) to the end of the specified linked list whose starting address is given in the first argument. Function addWorker is expected to **dynamically allocate memory space** (using **malloc** or **calloc**) for the new Worker and add the new Worker to the end of the linked list.

5. Define a function with the following prototype:

```
void removeWorker(ListPtr *listPtr, int id);
```

that searches for the Worker with the given id in the list. If the Worker is found, it removes the Worker node from the list and updates the pointers of its previous and next nodes in the list properly. You should use **free** function to release dynamically allocated memory of the removed Worker node.

6. Define a function with the following prototype:

```
void printWorkers(ListPtr workers);
```

which prints all the Worker elements in the list with the address given in the argument.

A sample output format is as follows. Id and name in the first line, followed by the age, dept and salary information for each worker in the given list.

```
================ 1111 George =============
Age:    25
Dept:   Finance
Salary: 3000
==========================================
================ 4444 Carla ==============
Age:    30
Dept:       Sales
Salary: 5000
==========================================
```

7. In the main function (**test.c**), test the correctness of your implementation.

   Create a pointer to the start of the linked list named WorkersList as shown below. It is initially set to NULL to indicate an empty list. Once an element is added, WorkersList would keep the address of the first element in the list.

```
ListPtr WorkersList = NULL;
```

   Add the following five workers to the list calling the addWorker function as follows.

```
addWorker(&WorkersList, 1111, "George", 25, 1, 3000);
addWorker(&WorkersList, 4444, "Carla", 30, 0, 5000);
addWorker(&WorkersList, 2222, "Annie", 27, 3, 6000);
addWorker(&WorkersList, 7777, "John", 41, 3, 7000);
addWorker(&WorkersList, 5555, "Robbert", 33, 2, 4000);
```

   These worker nodes must be kept as a linked list in memory in the order:

   Worker (with id:1111) → Worker (with id:4444) → Worker (with id:2222) → Worker (with id:7777) → Worker (with id:5555)

   Use printWorkers function to print the elements of the list as follows.

```
printWorkers(WorkersList);
```

   Use removeWorker function to Remove worker with id: 2222 the list as follows.

```
removeWorker(&WorkersList, 2222);
```

   Print the current list.

   Remove the worker with id: 1111 from the list.

   Print the current list.

   Add a new worker with id 3333 to the list, and then print the current list.