Homework 6a

ALGORITHMS

Addrecord

```
1  Define a record pointer called end.
2  Define a record pointer called temp.
3  Copy from the start variable the address it's
   pointing to, into variable end.
4  If the address stored in end is not NULL:
5    While the next field of the record being
     pointed to by end is not NULL:
6       Copy from the next field of the record
        being pointed to by end the address
        being pointed to into the variable end.
7  Allocate space for a struct record on the
   heap and store its address in the temp.
8  Copy from the uname variable its value into
   the name field of the record pointed to by
   temp.
9  Copy from the uyob variable its value into
   the yob field of the record pointed to by
   temp.
10 Copy from the uaddr variable its value into
   the addr field of the record pointed to by
   temp.
11 Copy from the utelno variable its value into
   the telno field of the record pointed to by
   temp.
12 Copy the NULL pointer into the next field of
   the record pointed to by temp.
13 If start is pointing to null:
14   Copy the address stored at temp into start
15 Else if the next field of the record being
   pointed to by start is NULL:
16   Copy the address stored at temp into the
     next field of start.
17 Else:
18  Copy the address pointed to by temp to next
    field of the record being pointed to by
    end.
```

```c
struct record *end;
struct record *temp;

end = *start;

if (end != NULL)
{
    while (end->next != NULL)
    {
        end = end->next;
    }
}

temp = (struct record *)malloc(sizeof
(struct record));
strcpy(temp->name, uname);
temp->yob = uyob;
strcpy(temp->addr, uaddr);
strcpy(temp->telno, utelno);
temp->next = NULL;

if (*start == NULL)
{
    *start = temp;
}
else if ((**start).next == NULL)
{
    (**start).next = temp;
}
else
{
    end->next = temp;
}
```

Delete Record

```
1 Define a record pointer named last.
2 Define a record pointer named current.
3 Define a short named match.
4 Copy the value of 0 into match.
5 Copy from the start variable the address being pointed to into the current variable.
6 While the address stored in current is not NULL:
7   If the string stored in the name field of the record pointed to by current and uname are equal:
8     Copy the value of 1 into match.
9     If the address pointed to by start and the address pointed to by current are equal:
10      Copy from the next field of the record pointed to by current into the next field of the record being pointed to by start.
11    Else:
12      Copy from the next field of the record pointed to by current the into the next field of the record pointed to by last.
13  Copy from the current variable the address being pointed to into the address pointed to by last.
14  Copy from the next field of the record being pointed to by current into the address being pointed to by current.
15  If the value of match is not 0:
16    Delete the record whose address is in last.
17    Copy the value of 0 into match.
```

```c
struct record *last;
struct record *current;

short match = 0;
current = *start;

while (current != NULL)
{
    if (strcmp(current->name, uname) == 0)
    {
        match = 1;
        if (*start == current)
        {
            start = current->next;
        }
        else
        {
            last->next = current->next;
        }
    }
    last = current;
    current = current->next;

    if (match)
    {
        free(last);
        match = 0;
    }
}
```