Kadir Has University
Department of Computer Engineering
CE 241 – Programming Languages
Fall 2008 - Ahmet Ardal
Homework #8

1. Write a function that takes a pointer to a GradeNode list, an id and a grade value as input. Inside the function, do the following:
- Dynamically create a new GradeNode object,
- Copy grade and id parameters of the function into the newly created GradeNode object,
- Finally append the new GradeNode object to the end of the list.

Function Prototype:
```
void appendGrade(GradeNode *pHead, int id, int grade);
```

2. Write a function that takes a pointer to a GradeNode list and an id as input. The function should search the list for a GradeNode object whose id member is equal to id parameter that passed to the function by the caller. If the function finds a corresponding GradeNode object, it should return its address, otherwise it should return NULL.

Function Prototype:
```
GradeNode *findGradeById(const GradeNode *pHead, int id);
```

3. Write a function that takes a pointer to a GradeNode list as input and prints id and grade members of each GradeNode object in the list.

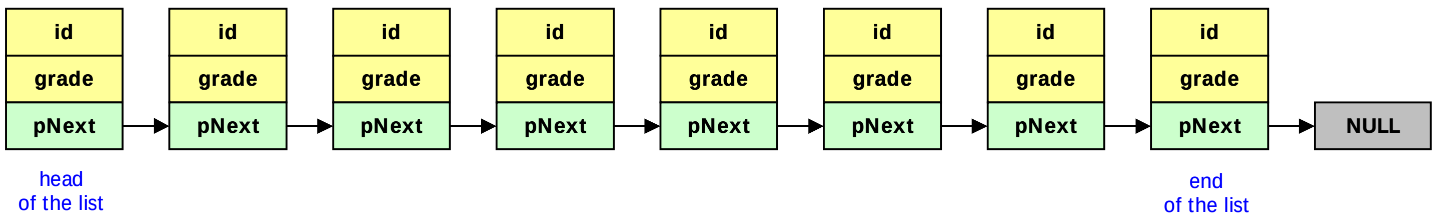
Function Prototype:
```
void printGradeList(const GradeNode *pHead);
```

4. Write a function that takes a pointer to a GradeNode list as input and deletes the first item of the list. The function should return the address of the pNext pointer of that deleted GradeNode item.

Function Prototype:
```
GradeNode *deleteFirstGrade(GradeNode *pHead);
```

Notes:

- Visualization of a sample GradeNode list that consists of individual GradeNode objects:



- Definition of the GradeNode structure:

```
struct GradeNode
{
    int id;
    int grade;
    GradeNode *pNext;
};
```

- Test code for the functions you are asked to implement in the preceding questions and the output of that test code for a sample execution:

```
#include <iostream>
using namespace std;

// struct declaration
struct GradeNode
{
    int id;
    int grade;
    GradeNode *pNext;
};

// function declarations
void        appendGrade(GradeNode *pHead, int id, int grade);
GradeNode *findGradeById(const GradeNode *pHead, int id);
void        printGradeList(const GradeNode *pHead);
GradeNode *deleteFirstGrade(GradeNode *pHead);

int main()
{
    GradeNode *pHead;  // pointer to the first item in the list

    // create the first node of the list
    pHead = new GradeNode;
    pHead->id = 77;
    pHead->grade = 70;
    pHead->pNext = NULL;

    // use appendGrade() function to add items to the list
    for (int i = 0; i < 5; ++i)
    {
        appendGrade(pHead, i, (i * i));
    }

    // use findGradeById() function to find a specific grade in the list
    GradeNode *pNode = findGradeById(pHead, 3);
    if (pNode == NULL)
    {
        cout << "cannot find grade..." << endl;
    }
    else
    {
        cout << "grade found: " << pNode->grade << endl;
    }

    // use printGradeList() function to print grades in the list
    cout << endl << "printing the list: " << endl;
    printGradeList(pHead);

    // use deleteFirstGrade() function to delete top two items from the list,
    // and print the list to see whether deletions were successful
    pHead = deleteFirstGrade(pHead);
    pHead = deleteFirstGrade(pHead);
    cout << endl << "printing the list after deletions: " << endl;
    printGradeList(pHead);

    // destroy the list and free all dynamically allocated memory
    for (GradeNode *pNode = pHead; pNode != NULL; )
    {
        GradeNode *pTmpNode = pNode->pNext;

        delete pNode;
        pNode = pTmpNode;
    }

    return 0;
}

// function implementations
// ...
```

Output from a sample execution of the test code above:

```
grade found: 9

printing the list:
id: 77, grade: 70
id: 0, grade: 0
id: 1, grade: 1
id: 2, grade: 4
id: 3, grade: 9
id: 4, grade: 16

printing the list after deletions:
id: 1, grade: 1
id: 2, grade: 4
id: 3, grade: 9
id: 4, grade: 16
Press any key to continue . . .
```