Program Purpose

This assignment is designed to give you practice working with structures, pointers, dynamic memory allocation, Unix, and linked lists.

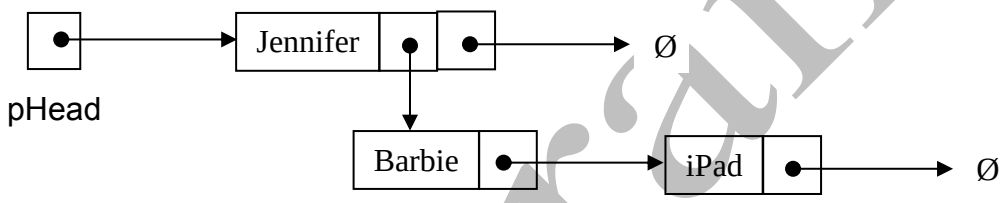
Always bring to class

1. Gaddis' book, How-to handouts from Canvas and your class notes.
2. This assignment sheet & the grade sheet for this lab already printed out.
3. USB Flash drive(s) or other storage media.

Mandatory Instructions

Santa Claus allegedly keeps lists of those who are naughty and those who are nice. On the naughty list are the names of those who will get coal in their stockings and therefore no gifts under the Christmas tree. On the nice list are those who will receive gifts. You will only implement the nice list in this assignment.

Each object on the nice list contains child's name (string) and a pointer to the first node of that child's list of gifts. Below are definitions for each child node on the nice list and each gift node for the gift list along with a visual representation of the nice list containing one child node, Jennifer, with two gifts on her list, Barbie, and iPad. Maintain child names in **alphabetic** order as they are inserted into the list. Do not maintain any order for the gifts.




```
// Structure definitions
//
struct Gift
{
	string desc;
	Gift*  pNext;
};
struct Nice
{
	string name;
	Nice*  pNext;
	Gift*  pGiftHead;
};
```

Any head pointers should be passed by **reference** so that means both pHead (pointer to the first node of the nice list) and pGiftHead (pointer to first node of the gifts) if there is a chance the address in that pointer may change.

```
// Prototypes
//
void createList(Nice* &);                  // Create the nice list
void appendGift(Gift* &, Gift*);           // Append to a gift list
void insertNice(Nice* &, Nice*);           // Insert into the nice list in alpha order by name
void printGifts(Nice*, string);            // Display a gift list
void printNiceList(Nice*);                 // Display the nice list
void destroyGiftList(Gift* &);             // Destroy a list of gifts
void destroyNiceList(Nice* &);             // Destroy the nice list
```

Data file

Data file prog4.txt will contain a list of names and gifts. Each line will contain a child first name followed by a list of blank separated gifts the child wants. Read each line from the data file (using getline) and parse out the child name and each of the gifts allocating either a Nice node or a Gift node as appropriate. Use string methods find() and substring() to parse out the portions of the input line. Examples of the use of find() and substring() are given below. Often programmers must research ways to accomplish a task. You will need to research find() and substring() functions and use them correctly.

```
string ss, s="hello", word;
int i;
i=s.find("e");                    // starts searching with pos 0 and returns pos when
                                  // pattern found. Here 1 would be returned
                                  // b/c pos of "e" is 1 within string "hello"
                                  // -1 is returned when no match if found

i=s.find("o",3);                  // 3 indicates the start pos is 3 and the search for "o" is
                                  // started there

ss=s.substring(0,2)               // returns "he" because start pos is 0 and
                                  // number of chars is 2 to return from the original string
```

Menu of Options

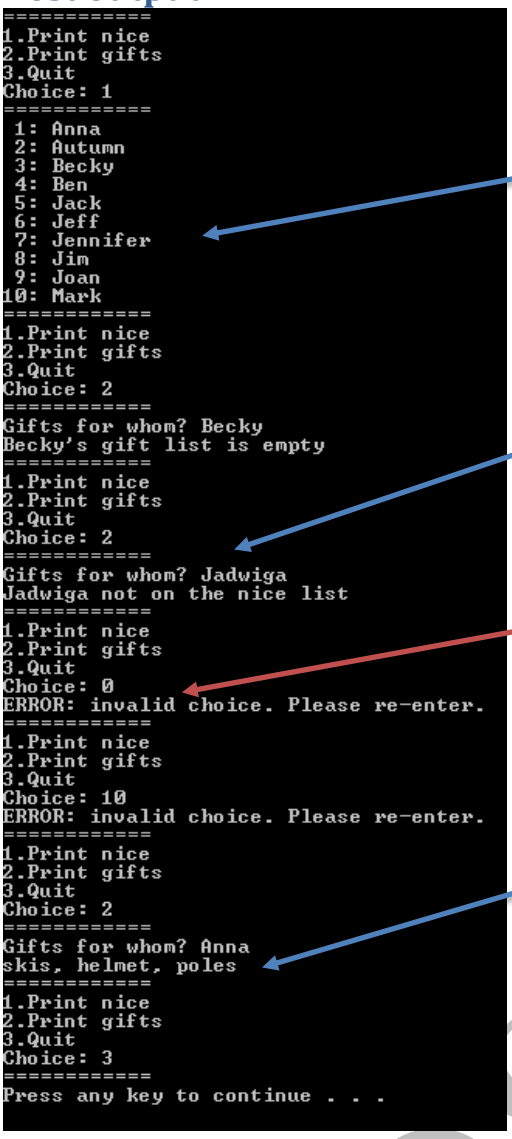
Program should present Santa the following menu of options:

```
1. Print nice
2. Print gifts
3. Quit
```

Utilize an enumerated type in your code that checks which option Santa selected. Validate Santa's input to make sure it is in the valid range 1 to 3. What if the user enters "three" instead of an integer? Your program should catch that, display an informative message and request valid input, i.e., do not accept as valid. You should handle the menu selection and validation in a separate function.

Option "print nice" will display all children's names one per line.
Option "print gifts" will prompt for name of child and then display the gifts for that child only. Only display gifts for a child that is found on the nice list, otherwise it is an error. Display appropriate message if child is on the list but has no gifts on the gift list. See example output below for additional formatting requirements.

Test output



Align ":" in the nice list display

A child may be on the list and have no gifts

Display ERROR for invalid menu choice

Separate gifts with comma and a space

Optional Instructions

Modify menu item (3) to delete a child from the nice list. Deallocate all memory for the removed child and his/her gifts.

Add menu item (4) to add a new child to the list in alpha order. Do not allow duplicates. Accept one line of input formatted as: name gift1 gift2 gift3…

Add menu item Quit which will now become (5)

What to turn in?

Place prog4.cpp and prog4.txt in your CS2020 BGLinux directory.

```
$ photo prog4.log
$ ls -l
```

CS2020, Instructor: Carlson

```
$ cat prog4.cpp
$ g++ prog4.cpp
$ .\a.out                // see screen shot for order of input; add more if you implemented bonus
$ Ctrl+d
$ exit
```

Upload grading sheet to Canvas!!!