**Week3\_Qn1**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/24/2023

Date Modified: 3/24/2023

Problem: Week 3 Question 1

Problem Description: inserts 10 random integers between 0 and 50 in order in a

linked list.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

typedef struct node {

int data;

struct node\* next;

} Node;

void insert(Node\*\* head, int data)

{

// Allocate memory for a new node and set its data value

Node\* new\_node = (Node\*) malloc(sizeof(Node));

new\_node->data = data;

new\_node->next = NULL;

// If the list is empty or the new node's data is less than the head's data, insert at the beginning

if (\*head == NULL || data < (\*head)->data)

{

new\_node->next = \*head;

\*head = new\_node;

}

else

{ // Otherwise, find the correct position to insert the new node and update pointers accordingly

Node\* current = \*head;

while (current->next != NULL && current->next->data < data)

{

current = current->next;

}

new\_node->next = current->next;

current->next = new\_node;

}

}

float average(Node\* head)

{

// Traverse the list, keeping track of the sum and count of elements

int sum = 0, count = 0;

while (head != NULL)

{

sum += head->data;

count++;

head = head->next;

}

// Calculate and return the average as a float

return (float) sum / count;

}

int main()

{

srand(time(NULL)); // Seed the random number generator with the current time

Node\* head = NULL; // Initialize the head pointer to NULL

for (int i = 0; i < 10; i++)

{

int data = rand() % 51; // Generate a random integer between 0 and 50

insert(&head, data); // Insert the new value in order into the linked list

}

printf("List elements: ");

Node\* current = head;

while (current != NULL)

{ // Traverse the list and print each element

printf("%d ", current->data);

current = current->next;

}

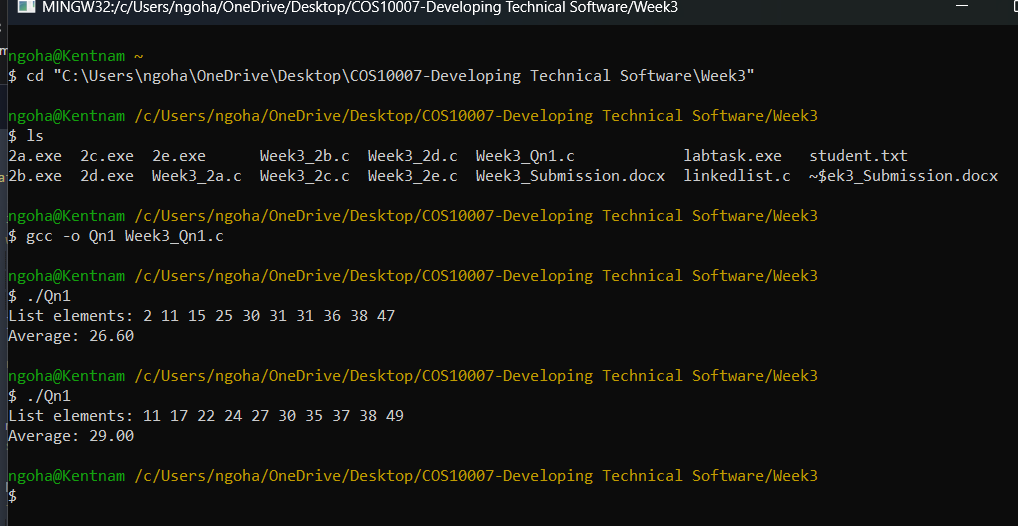
printf("\n");

printf("Average: %.2f\n", average(head)); // Calculate and print the average of the list

return 0;

}

OUTPUT FOR Qn1



**WEEK3\_2a.c**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/14/2023

Date Modified: 3/14/2023

Problem: Week 3 Question 2

Problem Description: Reading grade details of five students from a file and displaying it on screen

\*/

#include <stdio.h>

#include <stdlib.h>

struct personal

{

char studentName[20];

int studentId;

};

struct course

{

char courseName[20];

int courseId;

};

struct student

{

struct personal studentInfo;

struct course courseInfo;

struct student \*nextPtr;

};

typedef struct student Student;

typedef Student \*StudentPtr;

void printList(StudentPtr currentPtr)

{

puts("The list is \n");

while(currentPtr!=NULL)

{

printf("Student Name: %s \n",currentPtr->studentInfo.studentName);

printf("Course Name: %s \n", currentPtr->courseInfo.courseName);

printf("Student ID: %d \n",currentPtr->studentInfo.studentId);

printf("Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

int main()

{

FILE \*fp=fopen("student.txt","r");

if(fp==NULL)

{

printf("Error opening the file \n");

return -1;

}

StudentPtr startPtr=NULL;

startPtr=(Student\*)malloc(sizeof(Student));

fscanf(fp,"%s",startPtr->studentInfo.studentName);

fscanf(fp,"%s",startPtr->courseInfo.courseName);

fscanf(fp,"%d",&startPtr->studentInfo.studentId);

fscanf(fp,"%d",&startPtr->courseInfo.courseId);

startPtr->nextPtr=NULL;

printList(startPtr);

}

**OUTPUT FOR 2a**

**Text

Description automatically generated**

**WEEK3\_2b.c**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/14/2023

Date Modified: 3/14/2023

Problem: Week 3 Question 2b

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct personal

{

char studentName[20];

int studentId;

};

struct course

{

char courseName[20];

int courseId;

};

struct student

{

struct personal studentInfo;

struct course courseInfo;

struct student \*nextPtr;

};

typedef struct student Student;

typedef Student \*StudentPtr;

void printList(StudentPtr currentPtr)

{

puts("The list is \n");

while(currentPtr!=NULL)

{

printf("Student Name: %s \n",currentPtr->studentInfo.studentName);

printf("Course Name: %s \n", currentPtr->courseInfo.courseName);

printf("Student ID: %d \n",currentPtr->studentInfo.studentId);

printf("Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void insert(StudentPtr \*sPtr, char sName [], char cName[], int sID, int cID)

{

StudentPtr newPtr; /\* pointer to new node\*/

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

newPtr = malloc( sizeof( Student )); /\* create node\*/

if ( newPtr != NULL ) /\* is space available\*/

{

strcpy(newPtr->studentInfo.studentName,sName);

strcpy(newPtr->courseInfo.courseName, cName);

newPtr->studentInfo.studentId=sID;

newPtr->courseInfo.courseId=cID;

newPtr->nextPtr = NULL; /\* node does not link to another node\*/

previousPtr = NULL;

currentPtr = \*sPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && sID > currentPtr ->studentInfo.studentId)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr->nextPtr; /\* ... next node\*/

} /\* end while\*/

/\* insert new node at beginning of list\*/

if ( previousPtr == NULL )

{

newPtr->nextPtr = \*sPtr;

\*sPtr = newPtr;

} /\* end if\*/

else /\* insert new node between previousPtr and currentPtr\*/

{

previousPtr->nextPtr = newPtr;

newPtr->nextPtr = currentPtr;

} /\* end else\*/

} /\* end if\*/

else

{

printf( "No memory available.\n");

} /\* end else\*/

} /\* end function insert\*/

int main()

{

char sName[20], cName[20];

int sID, cID;

FILE \*fp=fopen("student.txt","r");

if(fp==NULL)

{

printf("Error opening the file \n");

return -1;

}

StudentPtr startPtr=NULL;

while(!feof(fp))

{

fscanf(fp,"%s",sName);

fscanf(fp,"%s",cName);

fscanf(fp,"%d",&sID);

fscanf(fp,"%d",&cID);

insert(&startPtr, sName,cName,sID,cID);

}

printList(startPtr);

}

**OUTPUT FOR 2b**

Text

Description automatically generated

**WEEK3\_2c.c**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/14/2023

Date Modified: 3/14/2023

Problem: Week 3 Question 2b

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct personal

{

char studentName[20];

int studentId;

};

struct course

{

char courseName[20];

int courseId;

};

struct student

{

struct personal studentInfo;

struct course courseInfo;

struct student \*nextPtr;

};

typedef struct student Student;

typedef Student \*StudentPtr;

void printList(StudentPtr currentPtr)

{

puts("The list is \n");

while(currentPtr!=NULL)

{

printf("Student Name: %s \n",currentPtr->studentInfo.studentName);

printf("Course Name: %s \n", currentPtr->courseInfo.courseName);

printf("Student ID: %d \n",currentPtr->studentInfo.studentId);

printf("Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void insert(StudentPtr \*sPtr, char sName [], char cName[], int sID, int cID)

{

StudentPtr newPtr; /\* pointer to new node\*/

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

newPtr = malloc( sizeof( Student )); /\* create node\*/

if ( newPtr != NULL ) /\* is space available\*/

{

strcpy(newPtr->studentInfo.studentName,sName);

strcpy(newPtr->courseInfo.courseName, cName);

newPtr->studentInfo.studentId=sID;

newPtr->courseInfo.courseId=cID;

newPtr->nextPtr = NULL; /\* node does not link to another node\*/

previousPtr = NULL;

currentPtr = \*sPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && sID > currentPtr ->studentInfo.studentId)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr->nextPtr; /\* ... next node\*/

} /\* end while\*/

/\* insert new node at beginning of list\*/

if ( previousPtr == NULL )

{

newPtr->nextPtr = \*sPtr;

\*sPtr = newPtr;

} /\* end if\*/

else /\* insert new node between previousPtr and currentPtr\*/

{

previousPtr->nextPtr = newPtr;

newPtr->nextPtr = currentPtr;

} /\* end else\*/

} /\* end if\*/

else

{

printf( "No memory available.\n");

} /\* end else\*/

} /\* end function insert\*/

void printfFile(StudentPtr currentPtr)

{

FILE \*fp=fopen("Output.txt","w+");

while(currentPtr!=NULL)

{

fprintf(fp,"Student Name: %s \n",currentPtr->studentInfo.studentName);

fprintf(fp,"Course Name: %s \n", currentPtr->courseInfo.courseName);

fprintf(fp,"Student ID: %d \n",currentPtr->studentInfo.studentId);

fprintf(fp,"Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

int main()

{

char sName[20], cName[20];

int sID, cID;

FILE \*fp=fopen("student.txt","r");

if(fp==NULL)

{

printf("Error opening the file \n");

return -1;

}

StudentPtr startPtr=NULL;

while(!feof(fp))

{

fscanf(fp,"%s",sName);

fscanf(fp,"%s",cName);

fscanf(fp,"%d",&sID);

fscanf(fp,"%d",&cID);

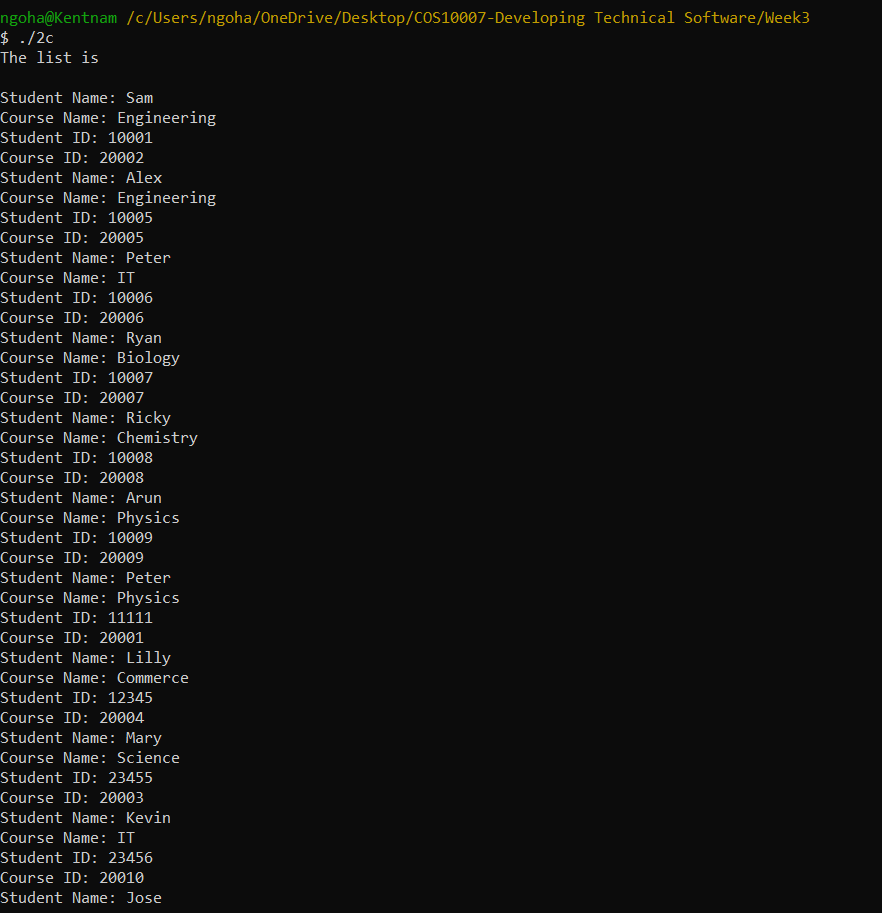
insert(&startPtr, sName,cName,sID,cID);

}

printList(startPtr);

}

**OUTPUT FOR 2c**

****

**WEEK3\_2d.c**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/14/2023

Date Modified: 3/14/2023

Problem: Week 3 Question 2d

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct personal

{

char studentName[20];

int studentId;

};

struct course

{

char courseName[20];

int courseId;

};

struct student

{

struct personal studentInfo;

struct course courseInfo;

struct student \*nextPtr;

};

typedef struct student Student;

typedef Student \*StudentPtr;

void printList(StudentPtr currentPtr)

{

puts("The list is \n");

while(currentPtr!=NULL)

{

printf("Student Name: %s \n",currentPtr->studentInfo.studentName);

printf("Course Name: %s \n", currentPtr->courseInfo.courseName);

printf("Student ID: %d \n",currentPtr->studentInfo.studentId);

printf("Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void printfFile(StudentPtr currentPtr)

{

FILE \*fp=fopen("Output.txt","w+");

while(currentPtr!=NULL)

{

fprintf(fp,"Student Name: %s \n",currentPtr->studentInfo.studentName);

fprintf(fp,"Course Name: %s \n", currentPtr->courseInfo.courseName);

fprintf(fp,"Student ID: %d \n",currentPtr->studentInfo.studentId);

fprintf(fp,"Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void insert(StudentPtr \*sPtr, char sName [], char cName[], int sID, int cID)

{

StudentPtr newPtr; /\* pointer to new node\*/

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

newPtr = malloc( sizeof( Student )); /\* create node\*/

if ( newPtr != NULL ) /\* is space available\*/

{

strcpy(newPtr->studentInfo.studentName,sName);

strcpy(newPtr->courseInfo.courseName, cName);

newPtr->studentInfo.studentId=sID;

newPtr->courseInfo.courseId=cID;

newPtr->nextPtr = NULL; /\* node does not link to another node\*/

previousPtr = NULL;

currentPtr = \*sPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && sID > currentPtr ->studentInfo.studentId)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr->nextPtr; /\* ... next node\*/

} /\* end while\*/

/\* insert new node at beginning of list\*/

if ( previousPtr == NULL )

{

newPtr->nextPtr = \*sPtr;

\*sPtr = newPtr;

} /\* end if\*/

else /\* insert new node between previousPtr and currentPtr\*/

{

previousPtr->nextPtr = newPtr;

newPtr->nextPtr = currentPtr;

} /\* end else\*/

} /\* end if\*/

else

{

printf( "No memory available.\n");

} /\* end else\*/

} /\* end function insert\*/

void delete(StudentPtr \*sPtr, char sName[])

{

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

//StudentPtr tempPtr=\*sPtr; /\* temporary node pointer\*/

/\* delete first node\*/

StudentPtr tempPtr=\*sPtr;

if ( strcmp(sName,(\*sPtr )-> studentInfo.studentName)==0)

{

\*sPtr = ( \*sPtr )->nextPtr; /\* de-thread the node\*/

free( tempPtr ); /\* free the de-threaded node\*/

} /\* end if\*/

else

{

previousPtr = \*sPtr;

currentPtr = ( \*sPtr )->nextPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && strcmp(currentPtr -> studentInfo.studentName,sName)!=0)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr -> nextPtr; /\*... next node\*/

} /\* end while\*/

/\* delete node at currentPtr\*/

if ( currentPtr != NULL )

{

tempPtr = currentPtr;

previousPtr->nextPtr = currentPtr->nextPtr;

free( tempPtr );

} /\* end if\*/

} /\* end else\*/

} /\* end function delete\*/

int main()

{

char sName[20], cName[20];

int sID, cID;

FILE \*fp=fopen("student.txt","r");

if(fp==NULL)

{

printf("Error opening the file \n");

return -1;

}

StudentPtr startPtr=NULL;

while(!feof(fp))

{

fscanf(fp,"%s",sName);

fscanf(fp,"%s",cName);

fscanf(fp,"%d",&sID);

fscanf(fp,"%d",&cID);

insert(&startPtr,sName,cName,sID,cID);

}

printList(startPtr);

//part d below

printf("Enter a name to delete \n");

scanf("%s", sName);

delete(&startPtr, sName);

printf("The list after deleting %s \n", sName);

printList(startPtr);

}

Text

Description automatically generated

**OUTPUT FOR 2d**

Text

Description automatically generated

**WEEK3\_2e**

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/14/2023

Date Modified: 3/14/2023

Problem: Week 3 Question 2d

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct personal

{

char studentName[20];

int studentId;

};

struct course

{

char courseName[20];

int courseId;

};

struct student

{

struct personal studentInfo;

struct course courseInfo;

struct student \*nextPtr;

};

typedef struct student Student;

typedef Student \*StudentPtr;

void printList(StudentPtr currentPtr)

{

puts("The list is \n");

while(currentPtr!=NULL)

{

printf("Student Name: %s \n",currentPtr->studentInfo.studentName);

printf("Course Name: %s \n", currentPtr->courseInfo.courseName);

printf("Student ID: %d \n",currentPtr->studentInfo.studentId);

printf("Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void printfFile(StudentPtr currentPtr)

{

FILE \*fp=fopen("Output.txt","w+");

while(currentPtr!=NULL)

{

fprintf(fp,"Student Name: %s \n",currentPtr->studentInfo.studentName);

fprintf(fp,"Course Name: %s \n", currentPtr->courseInfo.courseName);

fprintf(fp,"Student ID: %d \n",currentPtr->studentInfo.studentId);

fprintf(fp,"Course ID: %d \n",currentPtr->courseInfo.courseId);

currentPtr=currentPtr->nextPtr;

}

}

void insert(StudentPtr \*sPtr, char sName [], char cName[], int sID, int cID)

{

StudentPtr newPtr; /\* pointer to new node\*/

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

newPtr = malloc( sizeof( Student )); /\* create node\*/

if ( newPtr != NULL ) /\* is space available\*/

{

strcpy(newPtr->studentInfo.studentName,sName);

strcpy(newPtr->courseInfo.courseName, cName);

newPtr->studentInfo.studentId=sID;

newPtr->courseInfo.courseId=cID;

newPtr->nextPtr = NULL; /\* node does not link to another node\*/

previousPtr = NULL;

currentPtr = \*sPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && sID > currentPtr ->studentInfo.studentId)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr->nextPtr; /\* ... next node\*/

} /\* end while\*/

/\* insert new node at beginning of list\*/

if ( previousPtr == NULL )

{

newPtr->nextPtr = \*sPtr;

\*sPtr = newPtr;

} /\* end if\*/

else /\* insert new node between previousPtr and currentPtr\*/

{

previousPtr->nextPtr = newPtr;

newPtr->nextPtr = currentPtr;

} /\* end else\*/

} /\* end if\*/

else

{

printf( "No memory available.\n");

} /\* end else\*/

} /\* end function insert\*/

void delete(StudentPtr \*sPtr, char sName[])

{

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

//StudentPtr tempPtr=\*sPtr; /\* temporary node pointer\*/

/\* delete first node\*/

StudentPtr tempPtr=\*sPtr;

if ( strcmp(sName,(\*sPtr )-> studentInfo.studentName)==0)

{

\*sPtr = ( \*sPtr )->nextPtr; /\* de-thread the node\*/

free( tempPtr ); /\* free the de-threaded node\*/

} /\* end if\*/

else

{

previousPtr = \*sPtr;

currentPtr = ( \*sPtr )->nextPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && strcmp(currentPtr -> studentInfo.studentName,sName)!=0)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr -> nextPtr; /\*... next node\*/

} /\* end while\*/

/\* delete node at currentPtr\*/

if ( currentPtr != NULL )

{

tempPtr = currentPtr;

previousPtr->nextPtr = currentPtr->nextPtr;

free( tempPtr );

} /\* end if\*/

} /\* end else\*/

} /\* end function delete\*/

void update(StudentPtr \*sPtr, char sName[])

{

StudentPtr previousPtr; /\* pointer to previous node in list\*/

StudentPtr currentPtr; /\* pointer to current node in list\*/

char cName[20];

int cID, newSID;

/\* delete first node\*/

StudentPtr tempPtr=\*sPtr;

if ( strcmp(sName,(\*sPtr )-> studentInfo.studentName)==0)

{

\*sPtr = ( \*sPtr )->nextPtr; /\* de-thread the node\*/

strcpy(cName, (\*sPtr )->courseInfo.courseName);

cID=(\*sPtr)->courseInfo.courseId;

free( tempPtr ); /\* free the de-threaded node\*/

} /\* end if\*/

else

{

previousPtr = \*sPtr;

currentPtr = ( \*sPtr )->nextPtr;

/\* loop to find the correct location in the list\*/

while ( currentPtr != NULL && strcmp(currentPtr -> studentInfo.studentName,sName)!=0)

{

previousPtr = currentPtr; /\* walk to ...\*/

currentPtr = currentPtr -> nextPtr; /\*... next node\*/

} /\* end while\*/

/\* delete node at currentPtr\*/

if ( currentPtr != NULL )

{

tempPtr = currentPtr;

previousPtr->nextPtr = currentPtr->nextPtr;

strcpy(cName, currentPtr->courseInfo.courseName);

cID=currentPtr->courseInfo.courseId;

free( tempPtr );

} /\* end if\*/

} /\* end else\*/

printf("Enter a new student ID for %s\n",sName);

scanf("%d",&newSID);

insert(sPtr, sName, cName, newSID, cID);

}

int main()

{

char sName[20], cName[20];

int sID, cID;

FILE \*fp=fopen("student.txt","r");

if(fp==NULL)

{

printf("Error opening the file \n");

return -1;

}

StudentPtr startPtr=NULL;

while(!feof(fp))

{

fscanf(fp,"%s",sName);

fscanf(fp,"%s",cName);

fscanf(fp,"%d",&sID);

fscanf(fp,"%d",&cID);

insert(&startPtr,sName,cName,sID,cID);

}

printList(startPtr);

//part d below

printf("Enter a name to delete \n");

scanf("%s", sName);

delete(&startPtr, sName);

printf("The list after deleting %s \n", sName);

printList(startPtr);

//part e below

printf("Enter a name to update student's ID \n");

scanf("%s", sName);

update(&startPtr, sName);

printf("The list after modifying student ID of %c\n", sName);

printList(startPtr);

}

**OUTPUT FOR 2e**

**Text

Description automatically generated**

**Text

Description automatically generated**