**<lab 1>**

#include <stdio.h>

#include <stdlib.h>

#pragma warning(disable:4996)

struct STUDENT

{

char name[20];

float GPA;

int key;

struct STUDENT \*next;

};

typedef struct STUDENT Student;

typedef Student \*StudentPtr;

void Print(StudentPtr currentPtr);

void Insert(StudentPtr \*sPtr, char name[], float gpa, int num);

void freeList(StudentPtr head);

void main()

{

StudentPtr startPtr = NULL;

StudentPtr temp = NULL;

int i = 0, key = 0, j = 0;

float gpa = 0;

char name[20] = { 0, };

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

{

printf("Enter the name:\n");

for (i = 0; i < 20; i++)

{

scanf("%c", &name[i]);

if (name[i] == '\n')

{

name[i] = '\0';

break;

}

}

printf("Enter the key: ");

scanf("%d", &key);

printf("Enter the GPA: ");

scanf("%f", &gpa);

getchar();

Insert(&startPtr, name, gpa, key);

}

Print(startPtr);

freeList(startPtr);

system("pause");

}

void Insert(StudentPtr \*sPtr, char name[], float gpa, int key)

{

StudentPtr newPtr;

StudentPtr headPtr;

int i;

newPtr = malloc(sizeof(Student));

if (newPtr != NULL)

{

newPtr->GPA = gpa;

newPtr->key = key;

for (i = 0; i < 20; i++)

{

if (newPtr->name[i] == '\0')

{

break;

}

newPtr->name[i] = name[i];

}

headPtr = NULL;

newPtr->next = NULL;

headPtr = \*sPtr;

if (headPtr == NULL)

{

\*sPtr = newPtr;

headPtr = \*sPtr;

}

else{

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

{

if (headPtr == NULL)

{

printf("CurPtr isn't allocate");

break;

}

if (headPtr->next == NULL)

{

headPtr->next = newPtr;

break;

}

headPtr = headPtr->next;

}

}

}

else

{

printf("No memory available.\n");// 메모리 할당이 안되었을때

}

}

void Print(StudentPtr CurPtr)

{

int i;

if (CurPtr == NULL)

{

printf("List is empty.\n\n");

}

else{

while (CurPtr != NULL)

{

printf("name: ");

for (i = 0; i < 20; i++)

{

if (CurPtr->name[i] == '\0')

{

break;

}

printf("%c", CurPtr->name[i]);

}

printf("\tGPA = %f", CurPtr->GPA);

printf("\tkey = %d\n", CurPtr->key);

CurPtr = CurPtr->next;

}

}

}

void freeList(StudentPtr sPtr)

{

StudentPtr temp;

while (sPtr != NULL)

{

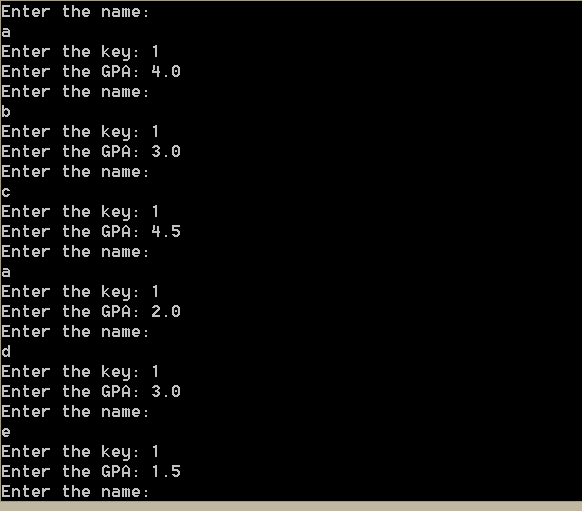
temp = sPtr;

sPtr = sPtr->next;

free(temp);

}

}



**<lab2>**

#include<stdio.h>

#pragma warning(disable:4996)

struct STUDENT

{

char name[20];

float GPA;

int key;

struct STUDENT \*next;

};

typedef struct STUDENT Student;

typedef Student \*StudentPtr;

void Print(StudentPtr CurPtr);

int InsertKey(StudentPtr \*sPtr, char name[], float gpa, int value);

void freeList(StudentPtr head);

int DeleteKey(StudentPtr \*sPtr, int value);

void main()

{

StudentPtr StartPtr = NULL;

StudentPtr temp = NULL;

int i = 0, key = 0, j = 0, inCheck = 0, deCheck = 0;

float gpa = 0;

char name[20] = { 0, };

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

{

printf("Enter the name\n");

for (j = 0; j < 20; j++)

{

scanf("%c", &name[j]);

if (name[j] == '\n')

{

name[j] = '\0';

break;

}

}

printf("Enter the GPA = ");

scanf("%f", &gpa);

printf("Enter the key = ");

scanf("%d", &key);

getchar();

if (key == -1)

{

break;

}

inCheck = InsertKey(&StartPtr, name, gpa, key

if (inCheck == 1)

{

printf("new node with key %d is inserted\n", key);

}

else if (inCheck == 2)

{

printf("key %d already exists\n", key);

}

}

for (i = 0; i < 5; i++)

{

printf("Enter the delete key = ");

scanf("%d", &key);

deCheck = DeleteKey(&StartPtr, key);// 1: 삭제할게 없음, 2: 삭제완료, 3: sPtr이 비어있는 경우

if (key == -1)

{

break;

}

if (deCheck == 1)

{

printf("key %d does not exist\n", key);

}

else if (deCheck == 2)

{

printf("node with key %d is deleted\n", key);

}

else if (deCheck == 3)

{

break;

}

}

print(StartPtr);

freeList(StartPtr);

system("pause");//

}

int InsertKey(StudentPtr \*sPtr, char name[], float gpa, int value) //Insertkey 함수

{

StudentPtr newPtr;

StudentPtr CurPtr;

StudentPtr PrePtr;

int i, check = 0;

newPtr = malloc(sizeof(Student));

if (newPtr != NULL)

{

newPtr->GPA = gpa;

newPtr->key = value;

for (i = 0; i < 20; i++)

{

if (newPtr->name[i] == '\0')

{

break;

}

newPtr->name[i] = name[i];

}

PrePtr = NULL;

CurPtr = NULL;

newPtr->next = NULL;

CurPtr = \*sPtr;

if (CurPtr == NULL)

{

\*sPtr = newPtr;

CurPtr = \*sPtr;

return 1;

}

else

{

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

{

if (CurPtr == NULL)

{

break;

}

if (CurPtr->key == newPtr->key)

{

check = 1;

free(newPtr);

return 2;

}

CurPtr = CurPtr->next;

}

if (check == 0)

{

CurPtr = \*sPtr;

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

{

if (CurPtr == NULL)

{

printf("CurPtr isn't allocatein in insert");

break;

}

if (CurPtr->key > newPtr->key)

{

if (PrePtr == NULL)

{

newPtr->next = \*sPtr;

\*sPtr = newPtr;

}

else

{

PrePtr->next = newPtr;

newPtr->next = CurPtr;

}

break;

}

if (CurPtr->next == NULL)

{

CurPtr->next = newPtr;

break;

}

PrePtr = CurPtr;

CurPtr = CurPtr->next;

}

return 1;

}

}

}

else

{

printf("No memory is available.\n");// 메모리 할당이 안 되었을때

}

return 0;

}

void Print(StudentPtr CurPtr)

{

int i;

if (CurPtr == NULL)

{

printf("List is empty.\n\n");

}

else

{

while (CurPtr != NULL)

{

printf("name = ");

for (i = 0; i < 20; i++)

{

if (CurPtr->name[i] == '\0')

{

break;

}

printf("%c", CurPtr->name[i]);

}

printf("\nGPA = %f", CurPtr->GPA);

printf("\nkey = %d\n", CurPtr->key);

CurPtr = CurPtr->next;

}

}

}

void freeList(StudentPtr sPtr)

{

StudentPtr temp;

while (sPtr != NULL)

{

temp = sPtr;

sPtr = sPtr->next;

free(temp);

}

}

int DeleteKey(StudentPtr \*sPtr, int value) //Deletekey함수

{

StudentPtr temp;

StudentPtr CurPtr;

StudentPtr PrePtr;

int i, check = 0;

if (\*sPtr == NULL)

{

return 3;

}

else if (value == (\*sPtr)->key)

{

temp = \*sPtr;

\*sPtr = (\*sPtr)->next;

free(temp);

return 2;

}

else

{

PrePtr = \*sPtr;

CurPtr = (\*sPtr)->next;

while (CurPtr != NULL&&CurPtr->key != value)

{

PrePtr = CurPtr;

CurPtr = CurPtr->next;

}

if (CurPtr != NULL)

{

temp = CurPtr;

PrePtr->next = CurPtr->next;

free(temp);

return 2;

}

else

{

return 1;

}

}

return 0;

}

