숙제 3

리스트 연산 (linked list)

순서

1. 프로그램 구조
2. Create 처리하기
3. Load 처리하기
4. Insert 처리하기
5. Delete 처리하기
6. Search 처리하기
7. Print 처리하기

0. 자료 구조

```
class sinfo {
          char name[8];
          char sex;
          char city[8];
          char dept[16];
          float gpa;
          int height;
          int weight;
public:
          void print ( ) {
                    printf("%s %c %s %s %.2f %d %d\n", this->name, this->sex,
                    this->city, this->dept, this->gpa, this->height, this->weight);
          }

          char *get_name ( ) {
                    return name;
          }
          void load ( char *str ) {
                    sscanf ( str, "%s %c %s %s %f %d %d", this->name, &this->sex,
                              this->city, this->dept, &this->gpa, &this->height,
                             &this->weight);
          }
```

0. 자료 구조

```
        void set ( char *tok2, char *tok3, char *tok4, char *tok5, char *tok6,
                  char *tok7, char *tok8 ) {
                  strcpy ( this->name, tok2 );
                  this->sex = tok3[0];
                  strcpy (this->city, tok4 );
                  strcpy (this->dept, tok5 );
                  this->gpa = (float) atof ( tok6 );
                  this->height = atoi ( tok7 );
                  this->weight = atoi ( tok8 );
        }
};
```

0. 자료 구조

```
class snode {
        sinfo item;
        snode *link;

public:
        char *get_name ( ) {
                return item.get_name ( );
        }
        void set ( sinfo nitem, snode *nlink ) {
                item = nitem;
                link = nlink;
        }
        void insert ( sinfo nitem );
        void print ( );
        void search ( char *tok );
        void remove ( char *tok );
};
```

0. 자료 구조

```
class hsnode {
        snode *link;

public:
        void process_create ( );
        void process_load ( char *fn );
        void process_insert ( sinfo nitem );
        void process_print ( );
        void process_delete ( char *tok );
        void process_search ( char *tok );
};
```

1. 프로그램 구조 (1)

```
int main ( )
{
   FILE *fp = fopen ( "input.txt", "r+t");
   char input[512];
   char tok1[32], tok2[32], tok3[32], tok4[32], tok5[32], tok6[32],
        tok7[32], tok8[32], tok9[32];


   hsnode *first = (hsnode *) malloc ( sizeof(hsnode) );
```

1. 프로그램 구조 (2)

```
        while ( fgets ( input, 512, fp ) != NULL ) {
                sscanf(input, "%s%s%s%s%s%s%s%s%s", tok1, tok2, tok3,
                        tok4, tok5, tok6, tok7, tok8, tok9);
                if ( strcmp ( tok1, "CREATE" ) == 0)
                        first->process_create ( );
                else if ( strcmp (tok1, "LOAD") == 0 )
                        first->process_load ( tok2 );
                else if ( strcmp(tok1, "PRINT") == 0)
                        first->process_print ( );
                else if (strcmp(tok1, "INSERT") == 0) {
                        sinfo nitem;
                        nitem.set ( tok2, tok3, tok4, tok5, tok6,
                                        tok7, tok8 );
                        first->process_insert ( nitem );
                }
                else if (strcmp(tok1, "DELETE") == 0)
                        first->process_delete ( tok2 );
                else if ( strcmp (tok1, "SEARCH") == 0 )
                        first->process_search ( tok2 );
                else
                        printf("%s is not a keyword.\n", tok1);
        }
```

1. 프로그램 구조 (3)

```
    fclose ( fp );
    return 0;
}
```

2. Create 처리하기

```
void hsnode::process_create ( )
{
        this->link = NULL;
}
```

3. Load 처리하기

```
void hsnode::process_load ( char *fn )
{
        FILE *fp2 = fopen ( fn, "r+t" );
        sinfo nitem;
        char str[512];
        while ( fgets ( str, 512, fp2 ) != NULL ) {
                nitem.load ( str );
                this->process_insert ( nitem );
        }

        fclose ( fp2 );
}
```

4. Insert 처리하기 (1)

```
void hsnode::process_insert ( sinfo nitem )
{
        //      degenrate case 1: inserting first node
        if ( this->link == NULL ) {                //      inserting first node
                snode *temp = (snode *) malloc ( sizeof(snode) );
                temp->set ( nitem, NULL );
                this->link = temp;
                return;
        }

        //      degenerate case 2: inserting before the first node
        if ( strcmp ( nitem.get_name ( ), this->link->get_name ( ) ) <= 0 ) {
                snode *temp = (snode *) malloc ( sizeof(snode) );
                temp->set ( nitem, this->link );
                this->link = temp;
                return;
        }

        this->link->insert ( nitem );
}
```

4. Insert 처리하기 (2)

```
void snode::insert ( sinfo nitem )
{
        snode *curr;

        for ( curr = this; curr->link != NULL; curr = curr->link ) {
                if ( strcmp ( curr->link->item.get_name ( ),
nitem.get_name( )) >= 0 )
                        break;
        }

        snode *nnode = (snode *) malloc ( sizeof(snode) );
        nnode->item = nitem;

        nnode->link = curr->link;
        curr->link = nnode;
}
```

5. Delete 처리하기

```
void hsnode::process_delete ( char *tok )
{
        if ( this->link != NULL )
                this->link->remove ( tok );
}
```

5. Delete 처리하기

```
void snode::remove ( char *tok )
{
        snode *curr;

        for ( curr = this; curr->link != NULL; curr = curr->link ) {
                if ( strcmp ( curr->link->item.get_name ( ), tok )==0) {
                        printf("Deleting %s.\n", tok);
                        curr->link = curr->link->link;
                        return;
                }
        }

        printf ("No %s in the list.\n", tok);
}
```

6. Search 처리하기

```
void hsnode::process_search ( char *tok )
{
        this->link->search ( tok );
}
```

6. Search 처리하기

```
void snode::search ( char *tok )
{
        snode *curr;

        for ( curr = this; curr != NULL; curr = curr->link ) {
                if ( strcmp ( curr->item.get_name ( ), tok ) == 0 ) {
                        printf("Found: ");
                        curr->item.print ( );
                        return;
                }
        }

        printf ("No %s in the list.\n", tok);
}
```

7. Print 처리하기

```
void snode::print ( )
{
        snode *curr;

        for ( curr = this; curr != NULL; curr = curr->link )
                curr->item.print ( );
}

void hsnode::process_print ( )
{
        if ( this->link == NULL )
                return;
        this->link->print ( );
}
```