# PA0 보고서

2016311821 한승하

저는 단어를 저장하기 위해 Insert시 시간을 최소로 줄이고자 Binary Tree 구조체를 사용하였습니다.

```
typedef struct node
{
        char* text;
        int num;
        struct node* left;
        struct node* right;
}Node;
```

다음과 같이 Tree를 만들 Struct를 선언해주었고

```
Node* NewNode(char* text,int len)
{
        int idx = 0;
        Node* new = (Node*)malloc(sizeof(Node));
        new->text = (char*)malloc(sizeof(char)*(len+1));
        for(idx=0;idx<len;idx++)
        {
                new->text[idx] = text[idx];
        }
        new->text[idx] = '\0';
        new->num = 1;
        new->left = NULL;
        new->right = NULL;
        return new;
}
```

NewNode로 새로운 Tree를 만들어주었고,

```
Node* insert(Node* Head, char* text,int len)
{
        int check;
        int idx = 0;
        Node* temp = (Node*)malloc(sizeof(Node));
        Node* find = Head;
        temp->text = (char*)malloc(sizeof(char)*(len+1));
        temp->num = 1;
        temp->left = NULL;
        temp->right = NULL;
        for(idx=0;idx<len;idx++)
        {
                temp->text[idx] = text[idx];
        }
        temp->text[idx] = '\0';
        while(1)
        {
                check = find_big(find->text,temp->text);
                if(check == 0)
                {
                        if(find->left == NULL)
                        {
                                find->left = temp;
                                return Head;
                        }
                        else find = find->left;
                }
                else if(check == 1)
                {
                        if(find->right == NULL)
                        {
                                find->right = temp;
                                        return Head;
                        }
                        else find = find->right;
                }
                else if(check == 2)
                {
                        find->num++;
                        free(temp);
                        return Head;
                }
        }
}
```

다음과 같은 Insert 함수를 통해 Binary 로 구현하였습니다.

이때 Binary의 기준은 정렬순서로 작으면 left, 크면 right에 저장되게 됩니다.

작다는 것은 정렬에서 우선이라는 뜻입니다.

```c
int find_big(char* src, char* dest)
{
        int idx = 0;
        while(1)
        {
                if(src[idx] == '\0' && dest[idx] == '\0') return 2;
                if(src[idx] == '\0') return 1;
                else if(dest[idx] == '\0') return 0;
                else
                {
                }
                if(src[idx] == dest[idx]) idx++;
                else
                {
                        if(src[idx] == 39) return 1;
                        if(dest[idx] == 39) return 0;
                        if(src[idx] == '-') return 1;
                        if(dest[idx] == '-') return 0;
                        if(src[idx]<='Z' && 'a'<=dest[idx])
                        {
                                if(src[idx] > dest[idx] - 32) return 0;
                                if(src[idx] < dest[idx] - 32) return 1;
                                if(src[idx] == dest[idx] - 32) return 1;
                        }
                        if('a'<=src[idx] && dest[idx]<='Z')
                        {
                                if(src[idx] -32 > dest[idx]) return 0;
                                if(src[idx] -32 < dest[idx]) return 1;
                                if(src[idx] -32 == dest[idx]) return 0;
                        }
                        if(src[idx]<dest[idx]) return 1;
                        else return 0;

                }
        }
        return 0;
}
```

다음과 같은 방법으로 크고 작음을 판단하게 되는데

이는 과제 설명에 나와있는 대로 ' ,- , 대문자, 소문자 순입니다.

이렇게 정렬되어 저장된 단어들은

```
void print_all(Node* Head)
{
        char* temp = (char*)malloc(sizeof(char)*(strlen(Head->text)+5));
        char* int_temp = (char*)malloc(sizeof(char)*5);
        if(Head->left != NULL) print_all(Head->left);
        sprintf(int_temp,"%d",Head->num);
        strcpy(temp,Head->text);
        strcat(temp," ");
        strcat(temp,int_temp);
        write(1,temp,strlen(temp));
        write(1,"\n",1);
        if(Head->right != NULL) print_all(Head->right);
}
```

다음과같이 작은 즉 우선권을 가진 단어들부터 출력하게 됩니다. 이때 write syscall을 사용하기 위해 spritnf와 strcpy, strcat을 사용하여 formet에 맞는 문자열을 만들어 출력하였습니다.

번외로

```
typedef struct shape
{
        char* newword;
        int len;
}shapeof;
```

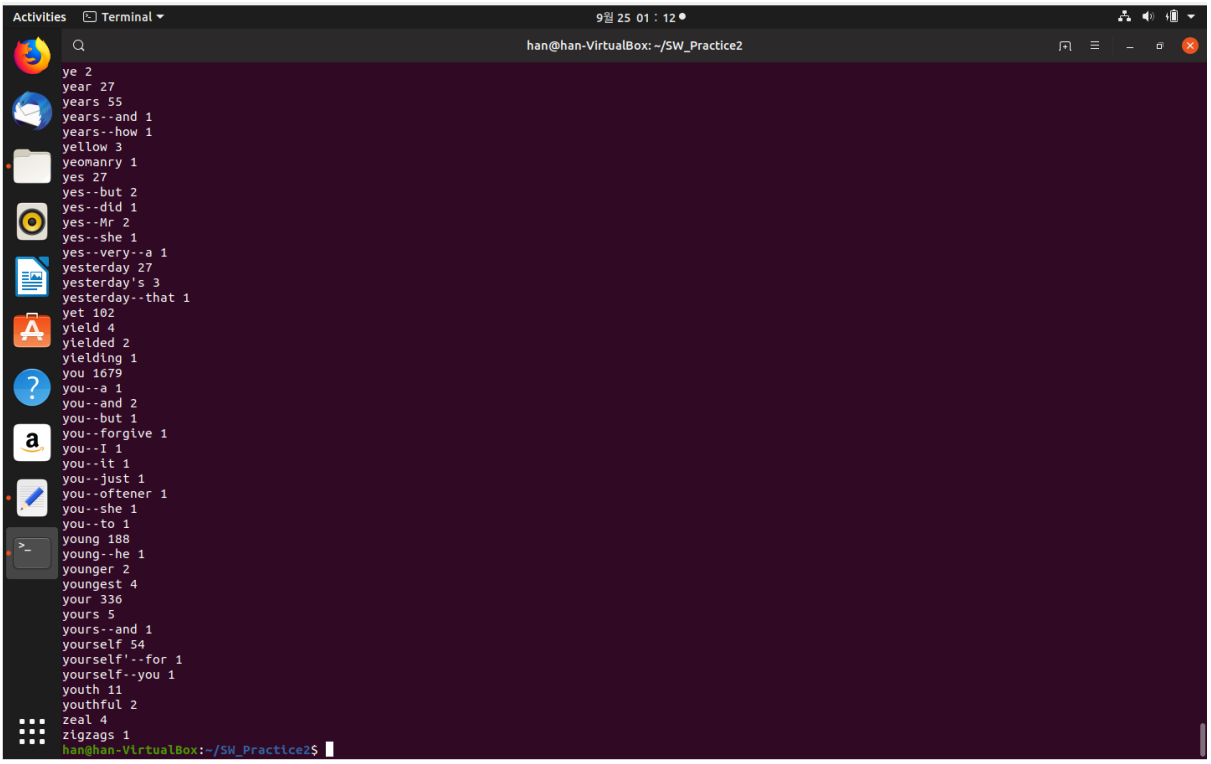
```
shapeof* shape(char* text,int len)
{
        shapeof* newshape = (shapeof*)malloc(sizeof(shapeof));
        int i;
        int idx = len+1;
        int checkif;
        char buffer[1];
        while(1)
        {
                buffer[0] = text[idx];
                checkif = check(buffer);
                if(checkif == 1)
                {
                        newshape->newword = (char*)malloc(sizeof(char)*(idx+1));
                        for(i=0;i<idx+1;i++)
                        {
                                newshape->newword[i] = text[i];
                        }
                        newshape->newword[idx+1] = '\0';
                        newshape->len = idx+1;
                        return newshape;
                }
                else
                {
                        text[idx] = '\0';
                        idx--;
                }
        }
}
```

다음은 단어의 뒤에 붙어있는 -와 '를 제거하고 단어의 정확한 length를 구하기 위한 함수와 구조체입니다.

마지막으로

```
int main(void)
{
        int idx = 0;
        int i;
        int first = 1;
        int state = 1;
        int checkif;
        char buffer[1];
        char *text = (char*)malloc(sizeof(char)*50);
        Node* Head;
        while(read(0,buffer,1))
        {
                shapeof *newshape;
                checkif = check(buffer);
                if(checkif)
                {
                        if(state == 1 && checkif == 1)
                        {
                                text[idx++] = buffer[0];
                                state = 0;
                        }
                        else if(state == 0)
                        {
                                text[idx++] = buffer[0];
                        }
                }
                else
                {
                        if(text[0] == '\0');
                        else
                        {
                        state = 1;
                        newshape = shape(text,idx);
                        if(first) Head = NewNode(newshape->newword,newshape->len);
                        else insert(Head,newshape->newword,newshape->len);
                        first = 0;
                        free(newshape->newword);
                        free(newshape);
                        for(i=0;i<idx;i++) text[i] = '\0';
                        idx = 0;
                        }
                }
        }
        print_all(Head);
        return 0;
```

위는 Main문이며




실행창은 다음과 같습니다.