

CP 111 Computer Programming

2/2554 Section B01/B02

Lab 12 solution

1. Problem name: Linked_List_Add_Student

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct s_node_tag {
    int ID;
    char name[21];
    int score;
    S_NODE *next;
};

int Add_Student(S_NODE *head,int ID,char *name,int score);

int main()
{
    S_NODE dummy_head,*ptr,*del_node;
    int s_id,s_score;
    char s_name[21];

    // set the linked list to be empty
    dummy_head.next=NULL;

    // read input data
    while(1)
    {
        scanf("%d %s %d",&s_id,s_name,&s_score);
        if(s_id==-1)
            break;
        Add_Student(&dummy_head,s_id,s_name,s_score);
    }

    // display nodes in linked list
    ptr=dummy_head.next;
    while(1)
    {
        if(ptr==NULL)
            break;
        printf("%d %s %d\n",ptr->ID,ptr->name,ptr->score);
        ptr=ptr->next;
    }

    // free memory allocated from linked list nodes
    ptr=dummy_head.next;
    while(ptr!=NULL)
    {
        del_node=ptr;
        ptr=ptr->next;
        free(del_node);
    }
    return 0;
}
```

```

int Add_Student(S_NODE *head,int ID,char *name,int score)
{
    S_NODE *new_s;

    new_s=(S_NODE *)malloc(sizeof(S_NODE));
    if(new_s==NULL)
        return -1;
    new_s->ID=ID;
    strcpy(new_s->name,name);
    new_s->score=score;
    new_s->next=head->next;
    head->next=new_s;
    return 0;
}

```

2. Problem name: Linked_List_Find_Student

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct s_node_tag {
    int ID;
    char name[21];
    int score;
    S_NODE *next;
};

int Add_Student(S_NODE *head,int ID,char *name,int score);
S_NODE *Find_Student(S_NODE *head,int q_id);

int main()
{
    S_NODE dummy_head,*ptr,*pMin;
    int s_id,s_score,q_id;
    char s_name[21];

    dummy_head.next=NULL;
    while(1)
    {
        scanf("%d %s %d",&s_id,s_name,&s_score);
        if(s_id==-1)
            break;

        Add_Student(&dummy_head,s_id,s_name,s_score);
    }

    scanf("%d",&q_id);

    ptr=Find_Student(&dummy_head,q_id);
    if(ptr!=NULL)
        printf("%d %s %d\n",ptr->ID,ptr->name,ptr->score);
    else
        printf("-1\n");
}

```

```

        //free memory allocated from linked list nodes
        ptr=dummy_head.next;
        while(ptr!=NULL)
        {
            del_node=ptr;
            ptr=ptr->next;
            free(del_node);
        }

        return 0;
    }

    int Add_Student(S_NODE *head,int ID,char *name,int score)
    // Similar to the previous problem

    S_NODE *Find_Student(S_NODE *head,int q_id)
    {
        S_NODE *ptr;
        ptr=head->next;
        while(ptr!=NULL)
        {
            if(ptr->ID==q_id)
                return(ptr);
            ptr=ptr->next;
        }
        return(NULL);
    }
}

```

3. Problem name: Linked_List_Delete_Student

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct s_node_tag S_NODE;
struct s_node_tag {
    int ID;
    char name[21];
    int score;
    S_NODE *next;
};

int Add_Student(S_NODE *head,int ID,char *name,int score);
S_NODE *Find_Student(S_NODE *head,int q_id);
int Delete_Student(S_NODE *head,int del_id);

int main()
{
    S_NODE dummy_head,*ptr,*pMin;
    int s_id,s_score,d_id,retval;
    char s_name[21];

    dummy_head.next=NULL;
    while(1)
    {
        scanf("%d %s %d",&s_id,s_name,&s_score);
        if(s_id==-1)
            break;
        Add_Student(&dummy_head,s_id,s_name,s_score);
    }
}

```

```

scanf("%d",&d_id);

Delete_Student(&dummy_head,d_id);

ptr=dummy_head.next;
while(1)
{
    if(ptr==NULL)
        break;
    printf("%d %s %d\n",ptr->ID,ptr->name,ptr->score);
    ptr=ptr->next;
}
//freememory allocated from linked list nodes
ptr=dummy_head.next;
while(ptr!=NULL)
{
    del_node=ptr;
    ptr=ptr->next;
    free(del_node);
}

return 0;
}

int Add_Student(S_NODE *head,int ID,char *name,int score)

S_NODE *Find_Student(S_NODE *head,int q_id)

int Delete_Student(S_NODE *head,int del_id)
{
    S_NODE *prev,*curr;

    prev=head;
    curr=head->next;
    while(curr!=NULL)
    {
        if(curr->ID==del_id)
        {
            prev->next=curr->next;
            free(curr);
            return 0;
        }
        prev=curr;
        curr=curr->next;
    }
    return -1;
}

```

4. Problem name: Max_occurrence_2_Linked_List

```
#include <stdio.h>
#include <stdlib.h>
typedef struct data_node_tag DATA_NODE;
struct data_node_tag {
    int data;
    int count;
    DATA_NODE *next;
};
int Update_Count(DATA_NODE *head, int data);
int main()
{
    DATA_NODE dummy_head,*ptr,*del_node;
    int retval,max_count,max_data,x;
    dummy_head.next=NULL;
    while(1)
    {
        scanf("%d",&x);
        if(x<0)
            break;
        Update_Count(&dummy_head,x);
    }
    ptr=dummy_head.next;
    max_count=0;
    max_data=-1;
    while(ptr!=NULL)
    {
        if(ptr->count > max_count)
        {
            max_count=ptr->count;
            max_data=ptr->data;
        }
        del_node=ptr;
        ptr=ptr->next;
        free(del_node);
    }
    printf("%d\n%d\n",max_data,max_count);
    return 0;
}
int Update_Count(DATA_NODE *head, int data)
{
    DATA_NODE *ptr,*new_node;
    ptr=head->next;
    while(ptr!=NULL)
    {
        if(ptr->data==data)
        {
            ptr->count=ptr->count+1;
            return 0;
        }
        ptr=ptr->next;
    }
    new_node=(DATA_NODE *)malloc(sizeof(DATA_NODE));
    if(new_node==NULL)
        return -1;
    new_node->data=data;
    new_node->count=1;
    new_node->next=head->next;
    head->next=new_node;
    return 0;
}
```