

CP 111 Computer Programming, 2/2554, Section B01/B02

Programming Homework 11 Solution

1. Problem name: Linked_List_Add_Student_Ascend

```
#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_Ascend(S_NODE *head, int ID, char *name, int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
int main()
{
    S_NODE dummy_head, *ptr, *pMin;
    int s_id, s_score, min_score;
    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_Ascend(&dummy_head, s_id, s_name, s_score);
    }
    Print_All_Students(&dummy_head);
    Free_All_Students(&dummy_head);
    return 0;
}

int Add_Student_Ascend(S_NODE *head, int id, char *name, int score)
{
    S_NODE *new_s, *prev, *curr;
    prev=head;
    curr=head->next;
    while(curr!=NULL && curr->ID < id)
    {
        prev=curr;
        curr=curr->next;
    }
    // insert new node after the node 'prev '
    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;
    prev->next=new_s;
    new_s->next=curr;
    return 0;
}
```

```

void Free_All_Students(S_NODE *head)
{
    S_NODE *del_node,*ptr;

    ptr=head->next;
    while(ptr!=NULL)
    {
        del_node=ptr;
        ptr=ptr->next;
        free(del_node);
    }
}

void Print_All_Students(S_NODE *head)
{
    S_NODE *ptr;
    ptr=head->next;
    while(ptr!=NULL)
    {
        printf("%d %s %d\n",ptr->ID, ptr->name, ptr->score);
        ptr=ptr->next;
    }
}

```

2. Problem name: Linked_List_Is_Subset

```

#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_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
int Linked_List_Is_Subset(S_NODE *head_sub, S_NODE *head_super);

int main()
{
    S_NODE dummy_head_1,dummy_head_2;
    int s_id,s_score,retval;
    char s_name[21];

    dummy_head_1.next=NULL;
    while(1)
    {
        scanf("%d %s %d",&s_id,s_name,&s_score);
        if(s_id==-1)
            break;
        Add_Student_Ascend(&dummy_head_1,s_id,s_name,s_score);
    }
    dummy_head_2.next=NULL;
    while(1)
    {
        scanf("%d %s %d",&s_id,s_name,&s_score);
        if(s_id==-1)
            break;
        Add_Student_Ascend(&dummy_head_2,s_id,s_name,s_score);
    }
    retval=Linked_List_Is_Subset(&dummy_head_2,&dummy_head_1);
}

```

```

        if(retval==0)
            printf("YES\n");
        else
            printf("NO\n");

        Free_All_Students(&dummy_head_1);

        Free_All_Students(&dummy_head_2);

        return 0;
    }

int Add_Student_Ascend(S_NODE *head,int id,char *name,int score)
{
}

void Free_All_Students(S_NODE *head)
{
}

void Print_All_Students(S_NODE *head)
{
}

int Linked_List_Is_Subset(S_NODE *head_sub, S_NODE *head_super)
{
    S_NODE *ptr1,*ptr2;
    int found;

    ptr1=head_sub->next;
    while(ptr1!=NULL)
    {
        ptr2=head_super->next;
        found=0;
        while(ptr2!=NULL)
        {
            if(strcmp(ptr1->name,ptr2->name)==0 &&
                ptr1->ID==ptr2->ID &&
                ptr1->score == ptr2->score)
            {
                found=1;
                break;
            }
            ptr2=ptr2->next;
        }
        if(found==0)
            return -1;
        ptr1=ptr1->next;
    }
    return 0;
}

```

3. Problem name: Linked_List_Update_Score

```

#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_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
void Recur_Print_Reverse(S_NODE *head);
int Update_Score(S_NODE *head,int s_id,int s_score);

```

```

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

    dummy_head.next=NULL;
    while(1)
    {
        scanf("%d %s",&s_id,s_name);
        if(s_id==-1)
            break;
        s_score=0;
        Add_Student_Ascend(&dummy_head,s_id,s_name,s_score);
    }
    while(1)
    {
        scanf("%d %d",&s_id,&s_score);
        if(s_id==-1)
            break;
        Update_Score(&dummy_head,s_id,s_score);
    }

    Print_All_Students(&dummy_head);

    Free_All_Students(&dummy_head);

    return 0;
}

```

```

int Update_Score(S_NODE *head,int s_id,int s_score)
{
    S_NODE *ptr;
    ptr=head->next;
    while(ptr!=NULL)
    {
        if(ptr->ID > s_id)
            return -1;

        if(ptr->ID == s_id)
        {
            ptr->score=ptr->score+s_score;
            return 0;
        }
        ptr=ptr->next;
    }
}

```

4. Problem name: Linked_List_Interlace

```
#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_Ascend(S_NODE *head,int ID,char *name,int score);
void Free_All_Students(S_NODE *head);
void Print_All_Students(S_NODE *head);
void Interlace(S_NODE *head,int id,char *name,int score);
int main()
{
    S_NODE dummy_head,*ptr;
    int s_id,s_score;
    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_Ascend(&dummy_head,s_id,s_name,s_score);
    }

    scanf("%d %s %d",&s_id,s_name,&s_score);
    Interlace(&dummy_head,s_id,s_name,s_score);
    Print_All_Students(&dummy_head);
    Free_All_Students(&dummy_head);
    return 0;
}

void Interlace(S_NODE *head,int i_id,char *i_name,int i_score)
{
    S_NODE *prev,*curr,*new_s;

    prev=head;
    curr=head->next;
    while(curr!=NULL)
    {
        new_s=(S_NODE *)malloc(sizeof(S_NODE));
        if(new_s==NULL)
            return;
        new_s->ID=i_id;
        new_s->score=i_score;
        strcpy(new_s->name,i_name);
        prev->next=new_s;
        new_s->next=curr;
        prev=curr;
        curr=curr->next;
    }
}
```

5. Problem name: Find_LT_Avg_Oldest_Student_2

```
#include <stdio.h>
typedef struct s_node_tag S_NODE;
struct s_node_tag {
    int ID;
    char fname[21];
    char lname[51];
    char gender;
    int age;
    double gpa;
    S_NODE *next;
};

int Add_Student_Ascend(S_NODE *head,int id,
                      char *fname,char *lname, char gender,int age,double gpa);
void Free_All_Students(S_NODE *head);

int main()
{
    S_NODE dummy_head;

    int i,N,sum,oldest_i,oldest_age,id,age;
    double avg,gpa;
    char fname[21],lname[51],gender;
    S_NODE *ptr,*oldest_ptr;

    dummy_head.next=NULL;
    while(1)
    {
        scanf("%d %s %s %c %d %lf",&id,fname,lname,&gender,&age,&gpa);
        if(id==0)
            break;
        Add_Student_Ascend(&dummy_head,id,fname,lname,gender,age,gpa);
    }

    // Calculate the mean
    sum=0;
    N=0;
    ptr=dummy_head.next;
    while(ptr!=NULL)
    {
        sum=sum+ptr->age;
        ptr=ptr->next;
        N=N+1;
    }
    avg=(double) sum/N;

    // Find the oldest student whose age < avg
    oldest_age=0;
    oldest_ptr=NULL;
    ptr=dummy_head.next;
    while(ptr!=NULL)
    {
        if(ptr->age<avg && ptr->age>oldest_age)
        {
            oldest_ptr=ptr;
            oldest_age=ptr->age;
        }
        ptr=ptr->next;
    }
}
```

```

printf("%.2f\n", avg);
printf("%d %s %s %c %d %.2f\n", oldest_ptr->ID,
                                         oldest_ptr->fname,
                                         oldest_ptr->lname,
                                         oldest_ptr->gender,
                                         oldest_ptr->age,
                                         oldest_ptr->gpa);

Free_All_Students(&dummy_head);

return 0;
}

int Add_Student_Ascend(S_NODE *head, int id, char *fname, char *lname,
                      char gender, int age, double gpa)
{
    S_NODE *new_s, *prev, *curr;

    prev=head;
    curr=head->next;

    while(curr!=NULL && curr->ID < id)
    {
        prev=curr;
        curr=curr->next;
    }
    // insert new node after the node 'prev '
    new_s=(S_NODE *)malloc(sizeof(S_NODE));
    if(new_s==NULL)
        return -1;

    new_s->ID=id;
    strcpy(new_s->fname, fname);
    strcpy(new_s->lname, lname);
    new_s->gender=gender;
    new_s->age=age;
    new_s->gpa=gpa;
    prev->next=new_s;
    new_s->next=curr;
    return 0;
}

void Free_All_Students(S_NODE *head)
{
    S_NODE *del_node, *ptr;

    ptr=head->next;
    while(ptr!=NULL)
    {
        del_node=ptr;
        ptr=ptr->next;
        free(del_node);
    }
}

```