Lab 6: Greedy Algorithm

Thực hành

Exercise 1:

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

int cmp1(const void* a, const void* b){
    return ((int*)a)[1] - ((int*)b)[1];
}

void actvSelect(int a[][2], int n){
    qsort(a, n, sizeof(a[0]), cmp1);
    int i, now = 0;
    printf("picked:\n");
    for(i=0;i<n;i++){
        if(a[i][0] >= now){
            printf("(%d,%d) ", a[i][0], a[i][1]);
            now = a[i][1];
        }
    }
    printf("\n");
}

int main(){
    int things[5][2] = {{1,3},{2,5},{4,7},{6,9},{8,10}};
    actvSelect(things, 5);
    return 0;
}
```

Exercise 2

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

int cmp2(const void* a, const void* b){
    return ((int*)a)[1] - ((int*)b)[1];
}

int late(int jbs[][2], int n){
    qsort(jbs, n, sizeof(jbs[0]), cmp2);
    int i, time = 0, mx = 0, done, late;
    for(i=0;i<n;i++){
        done = time + jbs[i][0];
        late = done - jbs[i][1];
        if(late<0) late = 0;
        if(late > mx) mx = late;
        time = done;
    }
    return mx;
}

int main(){
    int jobs[4][2] = {{3,4},{2,7},{1,5},{4,8}};
    printf("max late: %d\n", late(jobs, 4));
    return 0;
}
```

Exercise 3:

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

int cmp3(const void* a, const void* b){
    return ((int*)a)[0] - ((int*)b)[0];
}

int minRooms(int arr[][2], int n){
    qsort(arr, n, sizeof(arr[0]), cmp3);
    int rooms[100], rnum = 0;
    for(int i=0;i<n;i++){
        int put = 0;
        for(int j=0;j<rnum;j++){
            if(arr[i][0] >= rooms[j]){
                rooms[j] = arr[i][1];
                put = 1;
                break;
            }
        }
        if(!put){
            rooms[rnum++] = arr[i][1];
        }
    }
    return rnum;
}

int main(){
    int lec[3][2] = {{30,75},{0,50},{60,150}};
    printf("roomz: %d\n", minRooms(lec, 3));
    return 0;
}
```

Exercise 4:

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

```
#include<stdlib.h>

typedef struct {
    int id, profit, deadline;
} Job;

int cmp4(const void* a, const void* b){
    return ((Job*)b)->profit - ((Job*)a)->profit;
}

void maxProf(Job j[], int n){
    qsort(j, n, sizeof(Job), cmp4);
    int slot[100] = {-1}, total = 0;
    for(int i=0;i<n;i++) slot[i] = -1;

    printf("schedule: ");
    for(int i=0;i<n;i++){
        for(int j2 = (j[i].deadline<n?j[i].deadline:n)-1; j2 >= 0; j2--){
            if(slot[j2]==-1){
                slot[j2] = j[i].id;
                total += j[i].profit;
                printf("%d ", j[i].id);
                break;
            }
        }
    }
    printf("\nmoney: %d\n", total);
}

int main(){
    Job jbs[] = {{1,35,3},{2,30,4},{3,25,4},{4,20,2},{5,15,3},{6,12,1},{7,5,2}};
    maxProf(jbs, 7);
    return 0;
}
```

Exercise 5:

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

typedef struct {
    int val, wt;
    double ratio;
} Item;

int cmp5(const void* a, const void* b){
    double x = ((Item*)b)->ratio - ((Item*)a)->ratio;
    return (x > 0) - (x < 0);
}
```

```
double frack(Item it[], int n, int cap){
    for(int i=0;i<n;i++){
        it[i].ratio = (double)it[i].val / it[i].wt;
    }
    qsort(it, n, sizeof(Item), cmp5);

    double tot = 0.0;
    for(int i=0;i<n;i++){
        if(it[i].wt <= cap){
            tot += it[i].val;
            cap -= it[i].wt;
        }else{
            tot += it[i].val * ((double)cap / it[i].wt);
            break;
        }
    }
    return tot;
}

int main(){
    Item its[] = {{60,10},{100,20},{120,30}};
    printf("bag: %.2f\n", frack(its, 3, 50));
    return 0;
}
```