HackerRank For Work

MP6 B-2 > micah_chuim@yahoo.com

| | |
|---|---|
| **Full Name:** | Micah Nicole Chu Im |
| **Email:** | micah_chuim@yahoo.com |
| **Test Name:** | **MP6 B-2** |
| **Taken On:** | 13 Apr 2016 09:55:03 PHT |
| **Time Taken:** | 43 min 56 sec/ 90 min |
| **Invited by:** | Ryan |
| **Invited on:** | 12 Apr 2016 18:53:54 PHT |
| **Tags Score:** | |

Total Score
10/10

**Candidate Feedback:** sir ugh grabe lagi. nagarun sya perfectly sa command prompt then pagdito sa hackerrank kay may errors na hindi magets.

**Recruiter/Team Comments:**

*No Comments.*

| | Question Description | Time Taken | Score | Status |
|---|---|---|---|---|
| Q1 | I Am, First and Foremost, a Student > **Coding** | 33 min 14 sec | 10/ 10 | ✓ |

**QUESTION 1**

Correct Answer

Score 10

## I Am, First and Foremost, a Student > Coding

**QUESTION DESCRIPTION**

Create a structure that would represent a student. It should contain the following information:

- `hyphenated student number`
- `first name`
- `middle name`
- `last name`
- `program`
- `year level`

Assuming that we are keeping a record of all the students enrolled in UP Cebu, create another structure that contains an array of students (with 20 as capacity -- so much for keeping a record of all the students in UP Cebu) and of course, a size.

We should be able to do the following:

```
1.  enroll a student
2.  withdraw a student from the roll
3.  display all students
4.  search all students based on the year level
5.  search all students based on the course
6.  search all students based on the family name
7.  search a student based on the student number
```

ALL THESE SHOULD BE IMPLEMENTED AS FUNCTIONS (this means, apart from the main, you should have at least 7 functions).

The first input is the number of operations, say T, to be executed followed T inputs which are a combination of type of input (1-7, refer to the listing above) and the actual input.  The output is based on the input type.  If the input type is 1, 2, or 3, the output is the display of all the students in the record. The output for all other input types is still a display but based only on the search criteria.

The display should print all the information of a student exactly like the following:
Student Number: 1997-65336
Name: Dulaca, Ryan Ciriaco Madolin
Program: BS Math Computer Science
Year Level: 4

If there exists more than one student in the list, separate the display with a blank line exactly like the one below:
Student Number: 1997-65336
Name: Dulaca, Ryan Ciriaco Madolin
Program: BS Math Computer Science
Year Level: 4

Student Number: 2015-55038
Name: Noynay, Danny Boy Anislag
Program: BS Computer Science
Year Level: 1

Sample Input:
3
1
1997-65336
Ryan Ciriaco
Madolin
Dulaca
BS Math Computer Science
4
3
1
2015-55038
Danny Boy
Noynay
Anislag
BS Computer Science
1

Output:
Student Number: 1997-65336
Name: Dulaca, Ryan Ciriaco Madolin
Program: BS Math Computer Science
Year Level: 4

Student Number: 1997-65336
Name: Dulaca, Ryan Ciriaco Madolin
Program: BS Math Computer Science
Year Level: 4

Name: Dulaca, Ryan Ciriaco Madolin
Program: BS Math Computer Science
Year Level: 4

Student Number: 2015-55038
Name: Noynay, Danny Boy Anislag
Program: BS Computer Science
Year Level: 1

**CANDIDATE ANSWER**

Language used: **C**

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>
#define MAX 20

typedef struct{
    int yearLevel;
    char firstName[30],middleName[30],lastName[30],studentnumber[11], program[30];
}student;

typedef struct{
    int size;
    student record[20];
}list;


void init(list*);
int enroll(list*,student);
int withdraw(list*,char*);
void display(list);
int search(list ,char*);
void searchyearl(list,int);
void searchC(list,char*);
void searchfamily(list,char*);
void searchSN(list,char*);
int remo(list*,int);
int check(list ,char*);

int main() {
    int t,inputtype,c;
    student s;
    list l;

    init(&l);
    scanf("%d",&t);

    do{
        scanf("%d",&inputtype);

        if(inputtype==1){
            scanf(" %[^\n]s", s.studentnumber);
            scanf(" %[^\n]s", s.firstName);
            scanf(" %[^\n]s", s.middleName);
            scanf(" %[^\n]s", s.lastName);
            scanf(" %[^\n]s", s.program);
            scanf("%d", &s.yearLevel);

            c=check(l,s.studentnumber);

            if(c!=1)
            enroll(&l,s);

            else
            printf("THE STUDENT NUMBER %s ALREADY EXISTS.\n\n",s.studentnumber);

        display(l);
            //printf("\n");
        }
        if(inputtype==2){
             scanf(" %[^\n]s", s.studentnumber);
             withdraw(&l,s.studentnumber);

             display(l);
        }

        else if(inputtype==3){
```

```c
            display(l);
        }
        else if(inputtype==4){
            //search all students based on yearlevel
            scanf("%d",&s.yearLevel);
            searchyearl(l,s.yearLevel);
        }
        else if(inputtype==5){
            //search based on course
            scanf(" %[^\n]s",s.program);
            searchC(l,s.program);
        }
        else if(inputtype==6){
            //search all students based on the family name
            scanf(" %[^\n]s",s.lastName);
            searchfamily(l,s.lastName);
        }
        else if(inputtype==7){
            //search a student based on the student number
            scanf(" %[^\n]s",s.studentnumber);
            searchSN(l,s.studentnumber);
        }
        t--;
    }while(t>0);



    return 0;
}


void init(list* l){
    l->size = 0;
}

int enroll(list* l,student s){
    if(l->size == MAX)
        return 0;
    else{
        l->record[l->size++]=s;
        return 1;
    }
}

void display(list l){
    int i;
    for(i=0; i<l.size; i++){
        if(i>0)
        printf("\n");

        printf("Student Number: %s\n",l.record[i].studentnumber);
        printf("Name: %s, %s
%s\n",l.record[i].lastName,l.record[i].firstName,l.record[i].middleName);
        printf("Program: %s\n",l.record[i].program);
        printf("Year Level: %d\n",l.record[i].yearLevel);

    }
    printf("\n");
}

int withdraw(list* l,char * sn){
    int s= search(*l,sn);

    if(s!=-1){
        remo(l,s);
        return 1;
    }
    else
       return 0;
    }

int search(list l,char* t){
    int i;
    for(i=0; i<l.size; i++)
        if(strcmp(l.record[i].studentnumber,t)==0)
```

```c
            return i;
    return -1;
}

int remo(list* l,int pos){
    int i = pos;
    if(i<0 || i >= l->size)
        return 0;
    else{
        for(; i<l->size-1; i++)
            //strcpy(l->record[i],l->record[i+1]);
            l->record[i]=l->record[i+1];
        l->size--;
        return 1;
    }
}

void searchyearl(list l,int y){
    int i,flag=0;
    for(i=0; i<l.size; i++){
        if(l.record[i].yearLevel==y){
            if(i>0)
             printf("\n");

            printf("Student Number: %s\n",l.record[i].studentnumber);
            printf("Name: %s, %s
%s\n",l.record[i].lastName,l.record[i].firstName,l.record[i].middleName);
            printf("Program: %s\n",l.record[i].program);
            printf("Year Level: %d\n",l.record[i].yearLevel);

        flag=1;
    }

    }
    //printf("\n");

        if(flag==0){
        printf("NO STUDENT WITH YEAR LEVEL %d EXISTS.\n",y);
         printf("\n");
        }
}

void searchC(list l,char* t){
    int i,flag=0;
    for(i=0; i<l.size; i++){
        if(strcmp(l.record[i].program,t)==0){
            if(i>0)
            printf("\n");

            printf("Student Number: %s\n",l.record[i].studentnumber);
            printf("Name: %s, %s
%s\n",l.record[i].lastName,l.record[i].firstName,l.record[i].middleName);
            printf("Program: %s\n",l.record[i].program);
            printf("Year Level: %d\n",l.record[i].yearLevel);

        flag=1;
        }
    }
    //printf("\n");
        if(flag==0){
        printf("NO STUDENT WITH THE COURSE %s EXISTS.\n",t);
        //printf("\n");
        }
}

void searchfamily(list l,char* t){
    int i,flag=0;
    for(i=0; i<l.size; i++){
        if(strcmp(l.record[i].lastName,t)==0){
            if(i>0)
            printf("\n");

            printf("Student Number: %s\n",l.record[i].studentnumber);
            printf("Name: %s, %s
```

```
%s\n",l.record[i].lastName,l.record[i].firstName,l.record[i].middleName);
            printf("Program: %s\n",l.record[i].program);
            printf("Year Level: %d\n",l.record[i].yearLevel);

        flag=1;
        }
    }
    //printf("\n");
        if(flag==0){
        printf("NO STUDENT WITH THE FAMILY NAME %s EXISTS.\n",t);
       // printf("\n");
        }
}

void searchSN(list l,char* t){
    int i,flag=0;
    for(i=0; i<l.size; i++){
        if(strcmp(l.record[i].studentnumber,t)==0){
            if(i>0)
            printf("\n");

            printf("Student Number: %s\n",l.record[i].studentnumber);
            printf("Name: %s, %s
%s\n",l.record[i].lastName,l.record[i].firstName,l.record[i].middleName);
            printf("Program: %s\n",l.record[i].program);
            printf("Year Level: %d\n",l.record[i].yearLevel);


        flag=1;
        }
    }
    //printf("\n");
        if(flag==0){
        printf("NO SUCH STUDENT WITH STUDENT NUMBER %s EXISTS.\n",t);
        //printf("\n");
        }
}

int check(list l,char* t){
    int i;
    for(i=0; i<l.size; i++)
        if(strcmp(l.record[i].studentnumber,t)==0)
            return 1;
    return -1;
}



////enroll a student
//withdraw a student from the roll
//display all students
//search all students based on the year level
//search all students based on the course
//search all students based on the family name
//search a student based on the student number
```

| TESTCASE | TYPE | STATUS | SCORE | TIME TAKEN | MEMORY USED |
|---|---|---|---|---|---|
| Testcase 0 | Easy | Success | 1 | 0.0 sec | 2.53 MB |
| Testcase 1 | Easy | Success | 2 | 0.0 sec | 2.29 MB |
| Testcase 2 | Easy | Success | 2 | 0.0 sec | 2.28 MB |
| Testcase 3 | Easy | Success | 1 | 0.0 sec | 2.28 MB |
| Testcase 4 | Easy | Success | 1 | 0.0 sec | 2.28 MB |
| Testcase 5 | Easy | Success | 1 | 0.0 sec | 2.28 MB |
| Testcase 6 | Easy | Success | 1 | 0.0 sec | 2.28 MB |
| Testcase 7 | Easy | Success | 1 | 0.0 sec | 2.29 MB |

No Comments