```c
/******************************************************************************
 *    askEvent.c
 *
 *        This module used for ask information and validate information
 *
 *    Created by Chawakorn Boonrin(Bright) ID : 3415
 *        1 DECEMBER 2017
 *
 ******************************************************************************
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "ghostBuster.h"

/************** LOCAL FUNCTIONS, don't declare in header file ****************/

/* LOCAL FUNCTION. Remove 'new line' from last character out.
 * ARGUMENT:
 *     string - String want to remove 'new line'.
 */
void cutLine(char *string)
    {
    char *pCut;    /* Get string after found <CR>. */
    pCut = strpbrk(string,"\r\n");
    if (pCut != NULL)
        *pCut = '\0';
    }

/* LOCAL FUNCTION. This function print error of date.
 * ARGUMENT:
 *     error - To know what error it is.
 */
void printErrDate(int error)
    {
    switch(error)
        {
        case ERR_LEN :      /* Error with form of input. */
            printf("#SYSTEM: Please input date in form 'dd-mm-yyyy hh:mm'.\n");
            break;
        case ERR_CHAR :     /* Error with characters or puntuation. */
            printf("#SYSTEM: Please input only number in date and time.\n");
            break;
        case ERR_FORM1 :    /* Error with date or time not in scope. */
            printf("#SYSTEM: Date or time doesn't exist.\n");
            break;
        case ERR_FORM2 :    /* Error with time of event ealier than 10 years. */
            printf("#SYSTEM: Time of event must be in the past within 10 years.\n");
            break;
        case ERR_FORM3 :    /* Error with time of investigate is before time of event. */
            printf("#SYSTEM: Time of investigate must be after time of event.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function print error of name.
 * ARGUMENT:
 *     error - To know what error it is.
 */
void printErrName(int error)
    {
    switch(error)
        {
        case ERR_CHAR :     /* Error with number or punctuation. */
            printf("#SYSTEM: Name must be character, \"'\", \".\", \"-\" or spacebar.\n");
            break;
        case ERR_FORM1 :    /* Error with first character. */
            printf("#SYSTEM: First character must be alphabet.\n");
            break;
        case ERR_FORM2 :    /* Error with double punctuation. */
            printf("#SYSTEM: Name couldn't have punctuation next to it.\n");
            break;
        case ERR_FORM3 :    /*Error with last character. */
            printf("#SYSTEM: Last character is punctuation.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function print error of phone.
 * ARGUMENT:
 *     error - Address of string that want to return it back.
 */
void printErrPhone(int error)
    {
    switch(error)
        {
        case ERR_LEN :      /* Error with length of phone number. */
            printf("#SYSTEM: Phone number must be between 8 - 11.\n");
            break;
        case ERR_CHAR :     /* Error with characters or anothor puntuation. */
            printf("#SYSTEM: All phone number must be digit or dash.\n");
            break;
        case ERR_FORM1 :    /* Error with city code number. */
            printf("#SYSTEM: City code number is invalid.\n");
            break;
        case ERR_FORM2 :    /* Error with more than one dash. */
            printf("#SYSTEM: Must have only one dash.\n");
            break;
        case ERR_FORM3 :    /* Error with position of dash. */
            printf("#SYSTEM: Dash must be third or forth position of phone number.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function print error of Type.
 * ARGUMENT:
 *     error - Address of string that want to return it back.
 */
void printErrType(int error)
    {
    switch(error)
        {
        case ERR_LEN :      /* Error with length. */
            printf("#SYSTEM: Type of event is too long.\n");
            break;
        case ERR_CHAR :     /* Error with characters not allowed. */
            printf("#SYSTEM: Type can be only 'W','V','Z','G','O' and comma(,).\n");
            break;
        case ERR_FORM1 :    /* Error with double punctuation. */
            printf("#SYSTEM: Comma can't next to each other.\n");
            break;
        case ERR_FORM2 :    /* Error with first character. */
            printf("#SYSTEM: First character must be alphabet with W,V,Z,G and O.\n");
            break;
        case ERR_FORM3 :    /*Error with last character. */
            printf("#SYSTEM: Last character is punctuation.\n");
            break;
        case ERR_FORM4 :    /*Error with same type. */
            printf("#SYSTEM: There are same type.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function print error of location (Latitude/Longitude).
 * ARGUMENT:
 *     error - To know what error it is.
 */
void printErrLocation(int error)
    {
    switch(error)
        {
        case ERR_CHAR:   /* Error with not be number or period. */
            printf("#SYSTEM: Location must be number or period.\n");
            break;
        case ERR_FORM1:  /* Error with period or length of string. */
            printf("#SYSTEM: Latitude must be in form nn.nnnn.\n");
            break;
        case ERR_FORM2:  /* Error with period or length of string. */
            printf("#SYSTEM: Longitude must be in form nn.nnnn or nnn.nnnn.\n");
            break;
        case ERR_FORM3:  /* Latitude is wrong. */
            printf("#SYSTEM: Latitude must be between 8.0000 - 22.0000.\n");
            break;
        case ERR_FORM4:  /* Longitude is wrong. */
            printf("#SYSTEM: Longitude must be between 98.0000 - 102.0000.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function will print error of event code.
 * ARGUMENT:
 *     error - To know what error it is.
 */
void printErrCode(int error)
    {
    switch(error)
        {
        case ERR_LEN :      /* Error form of event code. */
            printf("#SYSTEM: Event code must be in form yyyy-nnnn.\n");
            break;
        case ERR_CHAR :     /* Error number and dash. */
            printf("#SYSTEM: Please input only number or dash in event code.\n");
            break;
        case ERR_FORM1 :    /* Error year in the future. */
            printf("#SYSTEM: Year in event code mustn't be in the future.\n");
            break;
        case ERR_FORM2 :    /* Error in code. */
            printf("#SYSTEM: 'nnnn' must not more than 9999 or less than 1.\n");
            break;
        }
    }
```

```
/* LOCAL FUNCTION. This function will print error of event code.
 * ARGUMENT:
 *     error - To know what error it is.
 */
void printErrYear(int error)
    {
    switch(error)
        {
        case ERR_LEN :      /* Error form of event code. */
            printf("#SYSTEM: Event year is invalid, try again.\n");
            break;
        case ERR_CHAR :     /* Error number and dash. */
            printf("#SYSTEM: Please input only number in event year.\n");
            break;
        case ERR_FORM1 :
             printf("#SYSTEM: Event year must not be within the past 10 years ");
             printf("or in the future.\n");
             break;
        }
    }
```

```
/***************** PUBLIC FUNCTIONS, declare in header file ******************/

/* PUBLIC FUNCTION. Print event detail to terminal line.
 * ARGUMENT:
 *     event - Struct event that want to print
 */
void printEvent(EVENT_T event)
    {
    char results[3][LENGTH] = {"SUCCESS","FAILURE","UNKNOWN"}; /* Store results. */
    printf("Event Code: %04d-%04d\n", event.eventCode[0], event.eventCode[1]);
    if (event.result == 0)
        printf("    Event has been deleted\n");
    else
        {
        printf("    Time Event\t\t\t: %02d/%02d/%04d %02d:%02d\n",
                event.dateEvent.day,
                event.dateEvent.month,
                event.dateEvent.year,
                event.dateEvent.hour,
                event.dateEvent.minute);
        printf("    Name Person Reportor\t: %s\n", event.nameReport);
        printf("    Phone Number\t\t: %s\n", event.phoneReport);
        printf("    Type of Event\t\t: %s\n", event.typeEvent);
        printf("    Latitude\t\t\t: %07.04f\n", event.latitude);
        printf("    Longitude\t\t\t: %08.04f\n", event.longitude);
        printf("    Time Investigate\t\t: %02d/%02d/%04d %02d:%02d\n",
                event.dateInvest.day,
                event.dateInvest.month,
                event.dateInvest.year,
                event.dateInvest.hour,
                event.dateInvest.minute);
        printf("    Name Person Investigating\t: %s\n", event.nameInvest);
        printf("    Result\t\t\t: %s\n", results[event.result-1]);
        }
    printf("\n");
    return;
    }
```

```
/* PUBLIC FUNCTION. Ask user to input "date of event" or "date of investigate" and
 * validate it. If user input <CR>, return '0' back (That's mean user cancel
 * to input data). But if date is valid, return name and '1' back.
 * ARGUMENT:
 *     pDate      - Address of date structure that want to return it back.
 *     pDateEvent - Get Address date of event. If it 'NULL', means ask date Event.
 */
int askDate(DATE_T *pDate, DATE_T *pDateEvent)
    {
    DATE_T date;               /* Keep struct of date temporary. */
    char input[LENGTH] = {0}; /* Get input from the terminal. */
    int done = 0;             /* Get result after validate. */
    while(1)
        {
        memset(input, 0, sizeof(input));
        if (pDateEvent == NULL)
            printf("Enter event date (dd-mm-yyyy hh:mm): ");
        else if (pDateEvent != NULL)
            printf("Enter investigate date (dd-mm-yyyy hh:mm): ");
        fgets(input, sizeof(input), stdin);
        if (strlen(input) == 1)   /* If user input <CR>, that mean user cancel. */
            return 0;
        cutLine(input);
        done = checkDateStr(input); /* Check string input. */
        /* If string is correct, send to validate event date/investigate date. */
        if (done == CORRECT)
            {
            sscanf(input, "%d-%d-%d %d:%d", &date.day, &date.month, &date.year,
                                            &date.hour, &date.minute);
            done = checkDate(date, pDateEvent);
            }
        if (done == CORRECT) /* If date is correct, return date and '1' back. */
            {
            memcpy(pDate, &date, sizeof(DATE_T));
            return 1;
            }
        else if (done != CORRECT) /* If date isn't correct, print error. */
            printErrDate(done);
        }
    }
```

```
/* PUBLIC FUNCTION. Ask user to input name of report or investigate. Then validate
 * input. If user input <CR>, return '0' back (That's mean user cancel to input
 * data). But if name is valid, return name and '1' back.
 * ARGUMENT:
 *     pOutput - Address of string that want to return it back.
 *     select  - To know what user want to ask
 *                    If select is '1', will ask name report.
 *                    If select is '2', will ask name investigate.
 */
int askName(char *pOutput, int select)
    {
    char input[LENGTH] = {0}; /* Get input from the terminal. */
    int done = 0;             /* Get result after validate. */
    while (1)
        {
        memset(input, 0, sizeof(input));
        /* Select to choose ask name report or name investigate. */
        if (select == 1)
            printf("Enter name report: ");
        else if (select == 2)
            printf("Enter name investigate: ");
        fgets(input, sizeof(input), stdin);
        if (strlen(input) == 1)  /* If user input <CR>, that mean user cancel. */
            return 0;
        cutLine(input);  /* Cut return at the end of character. */
        done = checkName(input);
        if (done == CORRECT)  /* If done is correct, return name and result back. */
            {
            strcpy(pOutput,input);
            return 1;
            }
        else if (done != CORRECT)  /* Input isn't correct, print error. */
            printErrName(done);
        }
    }
```

```
/* PUBLIC FUNCTION. Ask user to input phone number and validate it. If user input
 * return, return '0' back (That's mean user cancel to input data). But if phone
 * number phone is valid, return phone number and '1' back.
 * ARGUMENT:
 *     pOutput - Address of string that want to return it back.
 */
int askPhone(char *pOutput)
    {
    char input[SHORTLEN] = {0};  /* Get input from terminal line. */
    int done = 0;                /* Get check after validate. */
    while(1)
        {
        memset(input, 0, sizeof(input));
        printf("Enter Phone Number: ");
        fgets(input, sizeof(input), stdin);
        cutLine(input);
        if (strlen(input) == 0) /* If user input <CR>,  return '0'. */
            return 0;
        else
            {
            done = checkPhone(input);
            /* If it's valid, copy 'phoneNumber' to 'pOutput' to return back. */
            if (done == CORRECT)
                {
                strcpy(pOutput, input);
                return 1;
                }
            /* If phone number is invalid, print error and ask again. */
            else if (done != CORRECT)
                printErrPhone(done);
            }
        }
    }
```

```
/* PUBLIC FUNCTION. Ask user to input type and validate it. If user input <CR>,
 * return '0' back (That's mean user cancel to input data). But if type is valid,
 * return phone number and '1' back.
 * ARGUMENT:
 *     pOutputType - Address of string that want to return it back.
 */
int askType(char *pOutput)
    {
    char input[SHORTLEN];  /* Get input from the terminal. */
    int done = 0;          /* This will receive result after validate. */
    while (1)
        {
        memset(input, 0, sizeof(input));
        printf("Enter Types of Event: ");
        fgets(input, sizeof(input), stdin); /* Get type from user. */
        cutLine(input);
        if (strlen(input) == 0)  /* If user input <CR>, cancel to ask. */
            return 0;
        done = checkType(input);  /* Call function to check length. */
        if (done == CORRECT)      /* Input is correct. */
            {
            strcpy(pOutput, input);  /* Copy type to output. */
            return 1;
            }
        else if (done != CORRECT)    /* Input isn't correct, print error. */
            printErrType(done);
        }
    }
```

```
/* PUBLIC FUNCTION. Ask user to input latitude or longitude and validate it.
 * If user input <CR>, return '0' back (That's mean user cancel to input data).
 * But if latitude or longitude is valid, return name and '1' back.
 * ARGUMENT:
 *     pOutput - Address of string that want to return it back.
 *     select  - To know what user want to ask.
 *                    If select is '1', ask latitude.
 *                    If select is '2', ask longitude.
 */
int askLocation(float *pOutput, int select)
    {
    char input[SHORTLEN] = {0};   /* Get input from terminal line. */
    int done = 0;                 /* Check error. */
    float location = 0;           /* For keep location in float value. */
    while(1)
        {
        memset(input, 0, sizeof(input));
        /* Select to ask which one. 1-Latitude, 2- Longitude. */
        if (select == 1)
            printf("Enter latitude: ");
        else if (select == 2)
            printf("Enter longitude: ");
        fgets(input,sizeof(input),stdin);
        cutLine(input);
        if (strlen(input) == 0) /* If user input <CR>, cancel to ask. */
            return 0;
        done = checkStrLocation(input, select);
        if (done == CORRECT)
            {
            /* Get location in float value and send to validate. */
            sscanf(input,"%f",&location);
            done = checkLocation(location, select);
            }
        if (done == CORRECT)
            {
            /* If it is correct, return location and '1' back. */
            *pOutput = location;
            return 1;
            }
        else if (done != CORRECT)
            printErrLocation(done); /* If it's wrong, print error message. */
        }
    }
```

```c
/* PUBLIC FUNCTION. Ask users to input event code and validate it.
 * If the users input <CR>, return 0 back.
 * But if event code is valid, return event code and '1' back.
 * ARGUMENT:
 *     pOutput[] - Event code that users get.
 */
int askEventCode(int pOutput[])
    {
    char input[LENGTH] = {0}; /* Get input from the terminal. */
    int tempEventCode[2];     /* Keep event code temporary. */
    int done = 0;             /* Check error. */
    while(1)
        {
        memset(input, 0, sizeof(input));
        printf("Enter Event Code: ");
        fgets(input, sizeof(input), stdin);
        cutLine(input);
        if (strlen(input) == 0) /* If user input <CR>, cancel to ask event code. */
            return 0;
        done = checkEventCodeStr(input);
        if (done == CORRECT)
            { /* String of event code is correct, send to check event code. */
            sscanf(input, "%d-%d", &tempEventCode[0], &tempEventCode[1]);
            done = checkEventCode(tempEventCode);
            }
        if (done == CORRECT)
            { /* Event code is correct. */
            pOutput[0] = tempEventCode[0];
            pOutput[1] = tempEventCode[1];
            return 1;
            }
        else if (done != CORRECT) /* Event code is incorrect. */
            printErrCode(done);
        }
    }
```

```
/* PUBLIC FUNCTION. Ask users to input event year and validate it.
 * If the users input <CR>, return 0 back.
 * But if event year is valid, return event year and '1' back.
 * ARGUMENT:
 *     pEventYear - Event year that users get.
 */
int askEventYear(int *pEventYear)
    {
    char input[LENGTH] = {0}; /* Get input from the terminal. */
    int assumeCode[2] = {0};  /* Assume that event code send to validate. */
    int done = 0;             /* Keep value after validate int. */
    int i = 0;                /* Count loop. */
    while(1)
        {
        done = CORRECT;
        memset(input, 0, sizeof(input));
        printf("Enter Event year: ");
        fgets(input, sizeof(input), stdin);
        cutLine(input);
        if (strlen(input) == 0) /* If user input <CR>, stop to ask. */
            return 0;
        else if (strlen(input) != 4) /* Length of event year is incorrect. */
            done = ERR_LEN;
        for (i = 0; (i < strlen(input)) && (done == CORRECT); i ++)
            {
            if (!(isdigit(input[i])))
                { /* Event year is not digit. */
                done = ERR_CHAR;
                break;
                }
            }
        if(done == CORRECT)
            {
            assumeCode[1] = 1;
            sscanf(input, "%d", &assumeCode[0]);
            done = checkEventCode(assumeCode);
            }
        if ((done == CORRECT) || (done == ERR_FORM2))
            { /* Event year is correct. */
            *pEventYear = assumeCode[0];
            return 1;
            }
        printErrYear(done);
        }
    }
```

```
/* PUBLIC FUNCTION. Ask users to input result and validate it.
 * If the users input <CR>, return 0 back.
 * But if result is valid, return result and '1' back.
 * ARGUMENT:
 *     pResult   - Result that users get.
 */
int askResult(int * pResult)
    {
    char input[LENGTH] = {0}; /* Get input from the terminal. */
    int temp = 0;             /* Store value of result temporary. */
    while(1)
        {
        memset(input, 0, sizeof(input));
        printf("Enter result (1 = SUCCESS, 2 = FAILURE, 3 = UNKNOWN): ");
        fgets(input, sizeof(input), stdin);
        cutLine(input);
        if (strlen(input) == 0) /* If users input <CR>, stop to ask. */
            return 0;
        else if ((strlen(input) != 1) || !(isdigit(input[0]))) /* Result is incorrect. */
            printf("#SYSTEM: Error! Please input only number 1-3!\n");
        else
             {
             sscanf(input, "%d", &temp);
             if ((temp < 1) || (temp > 3)) /* Result is not between 1 and 3. */
             printf("#SYSTEM: Please select number 1-3.\n");
             else
             { /* Result is correct. */
             *pResult = temp;
             return 1;
             }
            }
        }
    }
```