Pham Tuan Minh
Digitally signed by Pham Tuan Minh
DN: cn=Pham Tuan Minh, email=phamtuanminh@gmail.com
Date: 2025.05.18 09:16:58 +07'00'

Bài tập thực hành
Cấu trúc dữ liệu và thuật toán
Ngăn xếp (A)

1. Bài 1

Cho khai báo kiểu của một nút trong danh sách liên kết như sau:

```
typedef struct _listnode {
  int num;
  struct _listnode *next;
} ListNode;
```

Cho khai báo kiểu của một danh sách liên kết như sau:

```
typedef struct _linkedlist {
  ListNode *head;
  int size;
} LinkedList;
```

Cho khai báo kiểu của một ngăn xếp như sau:

```
typedef struct _stack {
  LinkedList ll;
} Stack;
```

Cho khai báo của các hàm thực hiện các thao tác trên danh sách liên kết (đã cài đặt trong bài thực hành về danh sách liên kết) như sau:

```
void printList(ListNode *head);
```

Hiện ra màn hình giá trị của các phần tử trong danh sách liên kết trỏ bởi head. Nếu danh sách liên kết không có phần tử nào thì hiện thông báo "Danh sach lien ket khong co phan tu nao".

```
ListNode* findNode(ListNode *head, int i);
```

Trả về địa chỉ của node thứ i trong danh sách liên kết trỏ bởi head. Node đầu tiên của danh sách liên kết là node thứ 0.

void insertNode(ListNode **pHead, int index, int value);

Chèn một nút có giá trị value vào vị trí index trong danh sách liên kết trỏ bởi *pHead.

void removeNode(ListNode **ptrHead, int index);

Xóa nút ở vị trí index trong danh sách liên kết trỏ bởi *pHead.

Cho khai báo của các hàm thực hiện các thao tác trên ngăn xếp như sau:

void push(Stack *s, int item)

Đưa vào đỉnh ngăn xếp một phần tử có giá trị item.

int pop(Stack *s)

Lấy một phần tử ra khỏi đỉnh ngăn xếp. Giá trị trả về của hàm là giá trị của phần tử lấy ra.

int peek(Stack *s)

Trả về giá trị của nút ở đỉnh ngăn xếp, không tạo sự thay đổi gì trong ngăn xếp.

int isEmptyStack(Stack *s)

Trả về 1 nếu ngăn xếp rỗng, trả về 0 nếu ngăn xếp không rỗng.

1) Khai báo các cấu trúc dữ liệu và các hàm đã cho.

2) Hoàn thiện các hàm thao tác trên danh sách liên kết trong bài thực hành về danh sách liên kết.

3) Hoàn thiện hàm push()

```
void push(Stack *s, int item ) {
  insertNode( ___, ___, item);
  s->ll.size = ___ ;
}
```

4) Hoàn thiện hàm pop()

```
int pop(Stack *s) {
  int item;
  if (!isEmptyStack(s)) {
            item = ___ ;
            removeNode( ___, ___ );
            (s->ll).size = ___;
            return ___ ;
  }
  else {
            printf("Ngan xep rong");
            return NULL;
  }
}
```

5) Hoàn thiện hàm peek()

```
int peek(Stack *s) {
  if ( ___ )
            return ___ ;
  else {
            printf("Ngan xep rong");
            return NULL;
  }
}
```

6) Hoàn thiện hàm isEmptyStack()

```
int isEmptyStack(Stack *s) {
  if ( ___ ) return 1;
  return 0;
}
```

7) Hoàn thiện hàm main() thực hiện các công việc lần lượt theo thứ tự sau

Tạo một ngăn xếp rỗng.

Thêm các phần tử 10, 20, 30 vào ngăn xếp.

Dùng hàm printList() để đưa ra danh sách các phần tử trong ngăn xếp.

Lấy một phần tử ra khỏi ngăn xếp.

Thêm phần tử 40 vào ngăn xếp.

Lấy lần lượt các phần tử trong ngăn xếp và đưa kết quả ra màn hình cho đến khi ngăn xếp rỗng.

2. Bài 2

Cho khai báo kiểu của một ngăn xếp sử dụng mảng tĩnh như sau:

```
#define MAX 10

typedef struct {
   int data[MAX];
   int top;
} Stack;
```

Cho khai báo của các hàm thao tác trên ngăn xếp như sau:

void initStack(Stack *s);
Khởi tạo ngăn xếp, đặt top về -1.

int isEmptyStack(Stack *s);
Trả về 1 nếu ngăn xếp rỗng (top == -1), ngược lại trả về 0.

void push(Stack *s, int value);
Đưa giá trị value vào đỉnh ngăn xếp. Nếu ngăn xếp đầy, in thông báo "Stack Overflow".

int pop(Stack *s);
Lấy phần tử ở đỉnh ngăn xếp ra ngoài. Nếu ngăn xếp rỗng, in "Stack Underflow" và trả về -1.

int peek(Stack *s);
Trả về giá trị phần tử ở đỉnh ngăn xếp mà không xóa nó. Nếu ngăn xếp rỗng, in "Stack is empty" và trả về -1.

1) Khai báo cấu trúc dữ liệu và các hàm như đã cho.

2) Hoàn thiện hàm khởi tạo ngăn xếp initStack()

```
void initStack(Stack *s) {
   s->top = ___;
}
```

3) Hoàn thiện hàm isEmptyStack()

```
int isEmptyStack(Stack *s) {
   return s->top == ___;
}
```

4) Hoàn thiện hàm push()

```
void push(Stack *s, int value) {
   if (s->top >= MAX - 1) {
      printf("Stack Overflow");
      return;
   }
   s->top++;
   s->data[ ___ ] = value;
}
```

5) Hoàn thiện hàm pop()

```
int pop(Stack *s) {
   if (isEmptyStack(s)) {
      printf("Stack Underflow");
      return ___;
   }
   return s->data[ (s->top)___ ];
}
```

6) Hoàn thiện hàm peek()

```
int peek(Stack *s) {
   if (isEmptyStack(s)) {
      printf("Stack is empty");
      return ___;
   }
   return s->data[ ___ ];
}
```

7) Viết chương trình main() để kiểm tra các thao tác sau:

Khởi tạo ngăn xếp

Đưa các phần tử 5, 10, 15 vào ngăn xếp

In phần tử ở đỉnh ngăn xếp

Lần lượt lấy tất cả phần tử ra khỏi ngăn xếp và in giá trị từng phần tử

Gọi pop và peek khi ngăn xếp đã rỗng để kiểm tra xử lý ngoại lệ