Bài tập 4-1

Start Assignment

Due May 19 by 11:59pm **Points** 0 **Submitting** a file upload
File Types c, pdf, png, and jpg

Bài 1: Bài tập Cài đặt CTDL ngăn xếp
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 đoạn chương trình dưới đây trong hàm main() để tạo một ngăn xếp rỗng:

```
Stack *s = malloc( ___ );
s->ll.size = ___ ;
s->ll.head= ___ ;
```

4) Trong hàm main(), dùng hàm printList() để đưa ra danh sách các phần tử trong ngăn xếp.

5) Hoàn thiện hàm push()

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

6) 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;
}
}
```

7) Hoàn thiện hàm peek()

```
int peek(Stack *s) {
   if ( ___ )
      return ___ ;
   else {
```

```
printf("Ngan xep rong");
return NULL;
}
}
```

8) Hoàn thiện hàm isEmptyStack()

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

9) Viết hàm main() để kiểm tra kết quả khi gọi các hàm trên.

Bài 2: Bài tập Cài đặt CTDL hàng đợi

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 hàng đợi như sau:

```
typedef struct _queue {
   LinkedList ll;
} Queue;
```

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 hàng đợi như sau:
void enqueue(Queue *q, int item)
Đưa vào hàng đợi một phần tử có giá trị item.

int dequeue(Queue *q)
Lấy một phần tử ra khỏi hàng đợi. Giá trị trả về của hàm là giá trị của phần tử lấy ra.

int peek(Queue *q)
Trả về giá trị của nút ở đầu hàng đợi, không tạo sự thay đổi gì trong hàng đợi.

int isEmptyQueue(Queue *q)
Trả về 1 nếu hàng đợi rỗng, trả về 0 nếu hàng đợi 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 đoạn chương trình dưới đây trong hàm main() để tạo một hàng đợi rỗng:

```
Queue *q = malloc( ___ );
q->ll.size = ___ ;
q->ll.head= ___ ;
```

4) Trong hàm main(), dùng hàm printList() để đưa ra danh sách các phần tử trong hàng đợi.

5) Hoàn thiện hàm enqueue()

```
void enqueue(Queue *q, int item) {
   insertNode( ___ , ___, item);
   q->ll.size = ___ ;
}
```

6) Hoàn thiện hàm dequeue()

```
int dequeue(Queue *q) {
   int item;
   if (q->ll.head != NULL) {
      item = ___ ;
      removeNode( ___, ___ );
      (q->ll).size = ___ ;
      return item;
   }
   else {
prinft("Hang doi rong");
return 0;
}
}
```

7) Hoàn thiện hàm peek()

```
int peek(Queue *q){
    if (q->ll.head != NULL)
        return ___ ;
    else {
prinft("Hang doi rong");
return 0;
}
}
```

8) Hoàn thiện hàm isEmptyQueue()

```
int isEmptyQueue(Queue *q){
    if  (  ___ ) return 1;
    return 0;
}
```

9) Viết hàm main() để kiểm tra kết quả khi gọi các hàm trên.