|  |  |
| --- | --- |
| BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC PHENIKAA**  ĐỀ SỐ: 3  *Đề thi gồm có 5 câu; 4 trang*  *Đề thi* ***không được*** *sử dụng tài liệu* | **ĐỀ THI KẾT THÚC HỌC PHẦN**  *(Đối với môn thi tự luận)*  **Học phần:** Cấu trúc dữ liệu và thuật toán  **Mã học phần:** CSE703006  **Ngày thi:** **Giờ thi**:  Thời gian làm bài: 90 phút  *(Không kể thời gian giao đề)* |

**Họ và tên sinh viên**: ………………………………. **Số báo danh**:……………………

**Câu 1 (2 điểm - CĐR 1.1)**

Cho cấu trúc dữ liệu ngăn xếp viết bằng ngôn ngữ C như sau:

typedef struct \_listnode {

int num;

struct \_listnode \*next;

} ListNode;

typedef struct \_linkedlist {

ListNode \*head;

int size;

} LinkedList;

typedef struct \_stack {

LinkedList ll;

} Stack;

Hàm insertNode thực hiện chèn giá trị value vào vị trí có chỉ số index trong danh sách liên kết ptrHead được khai báo như sau:

void insertNode(ListNode \*\*ptrHead, int index, int value)

Hàm removeNode thực hiện xóa nút có chỉ số index trong danh sách liên kết ptrHead được khai báo như sau:

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

a) (1 điểm) Hoàn thành hàm push thực hiện chèn giá trị value vào ngăn xếp dùng hàm insertNode()

void push(Stack \*s, int item ) {

insertNode( \_\_\_ , \_\_\_ , item);

s->ll.size++;

}

b) (1 điểm) Hoàn thành hàm pop thực hiện lấy một phần tử từ ngăn xếp dùng hàm removeNode().

int pop(Stack \*s) {

int item;

if (!isEmptyStack(s)) {

item = ((s->ll).head)->num;

removeNode(\_\_\_ , \_\_\_);

(s->ll).size--;

return item;

}

else {

printf("\nNgan xep rong");

return NULL;

}

}

**Câu 2 (2 điểm - CĐR 2.1, 2.2, 2.3, 2.4 có trọng số ngang nhau)**

a) (1 điểm) Vẽ cây nhị phân tìm kiếm (BST) tạo thành bằng việc chèn lần lượt các phần tử của dãy khóa sau đây (bắt đầu từ cây rỗng): 10, 6, 17, 25, 4, 29, 8, 20, 13.

b (1 điểm) Vẽ cây nhị phân tìm kiếm (BST) tạo thành sau khi loại bỏ nút có khóa bằng 10 ra khỏi cây thu được ở câu a).

**Câu 3 (2 điểm - CĐR 2.1, 2.2, 2.3, 2.4 có trọng số ngang nhau)**

Cho mảng k: k[1], k[2], ..., k[n] chứa dãy n số nguyên.

Cho khai báo của hàm sắp xếp dãy n số theo thứ tự tăng dần dùng giải thuật giải thuật sắp xếp thêm dần như sau:

void insert\_sort(int k[], int n);

a. (1 điểm) Hoàn thiện mã chương trình của hàm insert\_sort()

void insert\_sort(int k[], int n) {

int i, j, x;

k[0] = 0;

for (i = 2; i <= n; i++) {

x = \_\_\_ ;

j = \_\_\_ ;

while ( x \_\_\_ k[j]) {

k[ \_\_\_ ] = k[ \_\_\_ ];

j = \_\_\_ ;

}

k[j+1] = x;

}

}

b. (1 điểm) Cho biết kết quả mảng k sau khi chạy xong for với i = 2 và i =3 khi gọi insert\_sort(b, 7) với mảng b khai báo như sau

int b[8] = {0, 99, 66, 22, 55, 11, 88, 77 };

**Câu 4. (3 điểm - CĐR 2.1, 2.2, 2.3, 2.4 có trọng số ngang nhau)**

Cho cài đặt của giải thuật Heapsort thực hiện sắp xếp n+1 phần tử K[0..n] theo thứ tự tăng dần như sau:

void ADJUST(int K[], int i, int n) {

int KEY = K[i];

int j = 2 \* i + 1;

while (j <= n) {

if ((j < n) && (K[j] < K[j + 1]))

j = j + 1;

if (KEY > K[j]) {

K[(int) floor((j-1) / 2)] = KEY;

return;

}

K[(int) floor((j-1) / 2)] = K[j];

j = 2 \* j + 1;

}

K[(int) floor( (j-1) / 2)] = KEY;

}

void HeapSort(int K[], int n) {

int i;

// (A)

for (i = floor((n-1) / 2); i >= 0; i--) {

ADJUST(K, i, n);

}

// (B)

for (i = n - 1; i >= 0; i--) {

int tmp = K[0];

K[0] = K[i + 1];

K[i + 1] = tmp;

ADJUST(K, 0, i);

}

// (C)

}

a) (1 điểm) Giải thích mục đích của đoạn lệnh từ (A) tới (B), và mục đích của đoạn lệnh từ (B) tới (C) trong hàm HeapSort().

b) (1 điểm) Cho mảng K có 5 phần tử: int K[5] = {55, 22, 11, 10, 99}.

Vẽ cây biểu diễn mảng K và các giá trị của mảng K là kết quả sau khi thực hiện đoạn lệnh từ (A) tới (B) khi chạy HeapSort(K, 4).

c) (1 điểm) Vẽ cây biểu diễn mảng K và các giá trị của mảng K là kết quả sau khi thực hiện xong vòng lặp for (của đoạn lệnh từ (B) tới (C)) với i = 3 khi chạy HeapSort(K, 4).

**Câu 5. (1 điểm - CĐR 2.1, 2.2, 2.3, 2.4 có trọng số ngang nhau)**

Xem xét các thao tác Enqueue (thêm phần tử vào queue) và Dequeue (xóa phần tử khỏi queue) trên các hàng đợi với k là một biến toàn cục.  
MultiDequeue(Q){

m = k

while (Q is not empty and m > 0) {

Dequeue(Q)

m = m - 1

}

}  
Độ phức tạp của dãy tuần tự n lệnh MultiDequeue(Q) với Q rỗng là trường hợp nào trong các trường hợp sau. Giải thích.

1. Θ(n) (b) Θ(n+k) (c ) Θ(nk) (d ) Θ(n^2)

TRƯỞNG BỘ MÔN/KHOA GIẢNG VIÊN RA ĐỀ