BỘ XÂY DỰNG
TRƯỜNG ĐHXD MIỀN TÂY

ĐÁP ÁN – THANG ĐIỂM
KỲ THI KTHP HỌC KỲ II NĂM HỌC 2023-2024
Trình độ: ĐẠI HỌC; Ngày thi: 29/05/2024
Môn: CẤU TRÚC DỮ LIỆU

ĐÁP ÁN ĐỀ THI CHÍNH THỨC

(Đáp án - thang điểm gồm 08 trang)

Lưu ý: Chương trình thực thi in ra màn hình kết quả đúng sinh viên sẽ được nhận tối đa số điểm câu đó. Nếu chương trình thực thi lỗi chấm theo thang điểm code bên dưới.

Câu	Phần	Nội dung	Thang điểm

1		`#include <stdio.h>`
		`#include <stdlib.h>`
		
		`typedef struct {`
		`  int  MSSV;`
		`  char hoten[30];`
		`  char lop[30];`
		`  int  giai;`
		`} ElementType;`
		
		`typedef struct Node* NodeType;`
		
		`struct Node {`
		`  ElementType element;`
		`  NodeType  next;`
		`};`
		
		`typedef NodeType Position;`
		`typedef Position List;`
			0.5

		`void MakeNullList(List *header) {`
		`  (*header)=(NodeType)malloc(sizeof(struct Node));`
		`  (*header)->next= NULL;`
		`}`
		
		`int EmptyList(List L) {`
		`  return (L->next==NULL);`
		`}`
		
		`Position First(List L) {`
		`  return L;`
		`}`
		
		`Position EndList(List L) {`
		`  Position p;`
		`  p=First(L);`
		`  while (p->next!=NULL)`
		`  p=p->next;`
		`  return p;`
		`}`
		` Position Next(Position p, List L) {`
			0.5

Câu	Phần	Nội dung	Thang điểm

		return p->next;
		}
		void InsertList(ElementType x,Position p, List *L) {
		Position t;
		t=(NodeType)malloc(sizeof(struct Node));
		t->element=x;
		t->next=p->next;
		p->next=t;
		}	

		void DeleteList(Position p, List *L) {
		Position temp;
		if (p->next!=NULL) {
		temp=p->next;
		p->next=temp->next;
		free(temp);
		}
		}
		Position Find(int maso, List L) {
		Position p;
		int Found = 0;
		p = L;
		while ((p->next != NULL) && (Found == 0))
		if (p->next->element.MSSV == maso) Found = 1;
		else p = p->next;
		return p;
		}
		void Delete(int maso, List *L) {
		Position p;
		p = Find(maso, *L);
		if(p != EndList(*L))
		DeleteList(p, L);
		else
		printf("Khong thay sinh vien co ma so nay\n");
		}
			0.5

		Position LocateSortedList(int giai, List L) {
		Position p;
		p = L;
		while ((p->next != NULL) && (p->next->element.giai<giai))
		p = Next(p, L);
		return p;
		}
		
		void InsertSortedList(ElementType x, List *L) {
		Position p = LocateSortedList(x.giai, *L);
		InsertList(x, p, L);
		}
			0.5

Câu	Phần	Nội dung	Thang điểm

		ElementType Retrieve(Position p, List L) {
		  if (p->next!=NULL)
		  return p->next->element;
		}
		void Sort(List *L) {
		  Position p, q, smallest;
		  p = First(*L);
		
		  while (p->next != NULL) {
		  smallest = p;
		  q = Next(p, *L);
		  while (q->next != NULL) {
		  if (q->next->element.giai < smallest->next->element.giai)
		  smallest = q;
		  q = Next(q, *L);
		  }
		  ElementType x;
		  x = p->next->element;
		  p->next->element = smallest->next->element;
		  smallest->next->element = x;
		  p = Next(p, *L);
		  }
		}	

		void PrintList(List L) {
		  Position p;
		  ElementType x;
		  p = First(L);
		  printf("MSSV\tTEN\tLOP\tGIAI\n");
		  while (p != EndList(L)) {
		  x = Retrieve(p,L);
		
		  printf("%d\t%s\t%s\t%d\n",x.MSSV,x.hoten, x.lop, x.giai);
		  p = Next(p, L);
		  }
		  printf("\n");
		}
			0.5

		void ReadList(List *L) {
		  int i, n;
		  MakeNullList(L);
		  printf("So sinh vien co trong danh sach = ");
		  scanf("%d",&n);
		  for(i=1;i<=n;i++) {
		  ElementType x;
		  printf("Sinh vien thu %d: \n",i);
		  printf("Ma sinh vien: "); scanf("%d",&x.MSSV);
		  getchar();
		  printf("Lop: "); gets(x.lop);
			0.5

Câu	Phần	Nội dung	Thang điểm

		printf("Ho ten: "); gets(x.hoten);
		printf("Dat giai: "); scanf("%d",&x.giai);
		if(Find(x.MSSV, *L) == EndList(*L))
		InsertList(x,EndList(*L),L);
		else
		printf("Ma so sinh vien da ton tai\n");
		}
		}	

		main() {
		List L;
		ElementType x;
		int maso;
		ReadList(&L);
		printf("Danh sach sinh vien hien tai la:\n");
		PrintList(L);
		printf("Danh sach giai tu cao xuong thap:\n");
		Sort(&L);
		PrintList(L);
		printf("Ma so sinh vien can xoa: "); scanf("%d",
		&maso);
		Delete(maso, &L);
		printf("Danh sach sau khi xoa la:\n");
		PrintList(L);
		printf("Phan tu can them\n");
		printf("Ma so sinh vien: "); scanf("%d",&x.MSSV);
		getchar();
		printf("Ho ten sinh vien: "); gets(x.hoten);
		printf("Lop: "); gets(x.lop);
		printf("Dat giai: "); scanf("%d",&x.giai);
		InsertSortedList(x, &L);
		printf("Danh sach hien tai la:\n");
		PrintList(L);
		
		return 0;
		}
			1.0

2		#include<stdio.h>
		#include<conio.h>
		#include<cstdlib>
		#define Max 100
		typedef int ElementType;
		typedef struct Node
		{
		ElementType Element;
		Node* Next;
		};
		typedef Node* Position;
		typedef Position Stack;
		void MakeNull_Stack(Stack *Header){
		(*Header)=(Node*)malloc(sizeof(Node));
			0.5

Câu	Phần	Nội dung	Thang điểm

		(*Header)->Next= NULL;
		}
		int Empty_Stack(Stack S){
		return (S->Next == NULL);
		}
		ElementType Top(Stack S){
		if (! Empty_Stack(S))
		return S->Next->Element;
		}	

		void Push(ElementType x, Stack *S)
		{
		Position temp;
		temp = (Node*)malloc(sizeof(Node));
		temp->Element = x;
		temp->Next = (*S)->Next;
		(*S)->Next = temp;
		}
		void Pop(Stack *S){
		if (! Empty_Stack(*S)){
		Position temp;
		temp = (*S)->Next;
		(*S)->Next = temp->Next; free(temp);
		}
		}
			0.5

		void Read_Stack(Stack *S){ int i,N;
		ElementType x;
		printf("So phan tu danh sach N = ");
		scanf("%d",&N);
		for(i=1; i<=N; i++){
		printf("Phan tu thu %d: ", i);
		fflush(stdin);
		scanf("%d", &x);
		Push(x, S);
		}
		}
		void Print_Stack(Stack *S){
		while(! Empty_Stack(*S)) {
		printf("%d \t", Top(*S));
		Pop(S);
		}
		printf("\n");
		}
			0.5

		int main(){ Stack S; ElementType n, temp, Fibo; MakeNull_Stack(&S); Read_Stack(&S); Print_Stack(&S); printf("Nhap so thap phan n = "); scanf("%d",&n);	**0.5**

Câu	Phần	Nội dung	Thang điểm

		if(n == 0)
		printf("So nhi phan tuong ung: 0");
		else{
		temp = n; MakeNull_Stack(&S);
		while(temp >0){
		Push(temp % 2, &S);
		temp = temp/2;
		}
		printf("So nhi phan tuong ung: ");
		while(! Empty_Stack(S)){
		printf("%d", Top(S)); Pop(&S);
		}
		}
		printf("\n");
		getch();
		return 0;
		}	

3	a	#include <stdio.h>
		#include <stdlib.h>
		typedef int KeyType;
		typedef struct Node* NodeType;
		struct Node {
		KeyType key;
		NodeType left,right;
		};
		
		typedef NodeType Tree;
		void MakeNullTree(Tree *T) {
		(*T)=NULL;
		}
		int EmptyTree(Tree T) {
		return T==NULL;
		}
			0.5

	a	void InsertNode(KeyType x, Tree *Root) {
		if (*Root == NULL) {
		(*Root)=(NodeType)malloc(sizeof(struct Node));
		(*Root)->key = x;
		(*Root)->left = NULL;
		(*Root)->right = NULL;
		}
		else if (x < (*Root)->key) InsertNode(x, &((*Root)->left));
		else if (x > (*Root)->key) InsertNode(x, &((*Root)->right));
		}
		Tree LeftChild(Tree n) {
		if (n!=NULL)
		return n->left;
		else
			0.5

Câu	Phần	Nội dung	Thang điểm

		return NULL;
		}
		
		Tree RightChild(Tree n) {
		if (n!=NULL)
		return n->right;
		else
		return NULL;
		}
		int IsLeaf(Tree n) {
		if(n!=NULL)
		
		return(LeftChild(n)==NULL)&&(RightChild(n)==NULL);
		else
		return NULL;
		}	

	b	void PreOrder(Tree T) {
		if(T != NULL) {
		printf("%d ",T->key);
		PreOrder(LeftChild(T));
		PreOrder(RightChild(T));
		}
		}
			0.5

	b	void InOrder(Tree T) {
		if(T != NULL) {
		InOrder(LeftChild(T));
		printf("%d ",T->key);
		InOrder(RightChild(T));
		}
		}
			0.5

	b	void PostOrder(Tree T) {
		if (T!=NULL) {
		PostOrder(LeftChild(T));
		PostOrder(RightChild(T));
		printf("%d ",T->key);
		}
		}
			0.5

	c	int nb_nodes(Tree T) {
		if(EmptyTree(T))
		return 0;
		else
		return 1 + nb_nodes(LeftChild(T)) +
		nb_nodes(RightChild(T));
		}
			0.5

	c	main() { Tree tree, node; KeyType x; int i, n; MakeNullTree(&tree);	**1.0**

Câu	Phần	Nội dung	Thang điểm

		printf("So nut nhap vao: "); scanf("%d", &n);
		for(i=0; i<n; i++) {
		printf("Nhap vao nut thu %d: ",i+1);
		scanf("%d", &x);
		InsertNode(x, &tree);
		}
		printf("\n duyet tien tu :"); PreOrder(tree);
		printf("\n duyet trung tu :"); InOrder(tree);
		printf("\n duyet hau tu :"); PostOrder(tree);
		printf("\n so nut cua cay la %d \n", nb_nodes(tree));
		}	

TỔNG ĐIỂM			**10.0đ**