Bài tập thực hành
Cấu trúc dữ liệu và thuật toán
Con trỏ, mảng, danh sách liên kết

1. Bài 1

Hoàn thành đoạn chương trình dưới:

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int h[4]={4,5,6};
  printf("Gia tri cua h[0]=%d\n", ____);
  printf("Gia tri cua h[4]=%d\n", ____);
  printf("Dia chi cua h[0]=%p\n", ____); // Cach 1
  printf("Dia chi cua h[0]=%p\n", ____); // Cach 2
  printf("Dia chi cua h[1]=%p\n", ____); // Cach 1
  printf("Dia chi cua h[1]=%p\n", ____); // Cach 2
  printf("Gia tri cua h[1]=%d\n", ____); // Cach 1
  printf("Gia tri cua h[1]=%d\n", ____); // Cach 2

  return 0;
}
```

2. Bài 2

```
int ex5FindMax1()
{
  int i;
  printf("\nDUYET MANG: TIM GIA TRI LON NHAT\n");
  int index, max, numArray[10];
```

```
  max = -1;
  printf("Nhap 10 gia tri so vao mang numArray: \n");
  for (index = 0; index < 10; index++) {
     scanf("___", ___);
  }

  for (index = 0;   index < 10;    index++) {
          if (numArray[index] > max)
                  max = numArray[index];
  }
  printf("Gia tri lon nhat la %d.\n", ___);
}
```

3. Bài 3

1) Cho hàm print1DimArr() thực hiện đưa ra màn hình giá trị của các phần tử của mảng 1 chiều. Hoàn thiện hàm print1DimArr():

```
void  print1DimArr(int ar[], int n) {
}
```

2) Cho hàm print2DimArrA() thực hiện đưa ra màn hình giá trị của các phần tử của mảng 2 chiều. Hoàn thiện hàm print2DimArrA()

```
int print2DimArrA(int ar[][3], int m){
  int i, j;

  for (i = 0; i < m; i++) {
          for (j = 0; j < 3; j++) {
                  printf("%d\t", ___ );
          }
          printf("\n");
  }
}
```

3) Cho hàm print2DimArrB() thực hiện đưa ra màn hình giá trị của các phần tử của mảng 2 chiều. Hoàn thiện hàm print2DimArrB()

```
int print2DimArrB(int ar[][3], int m){
  int i, j;
  for (i = 0; i < m; i++) {
        print1DimArr( ___ , 3);
        printf("\n");
  }
}
```

4. Thao tác trên danh sách liên kết

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 của các hàm thực hiện các thao tác trên 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.

1) Hoàn thiện đoạn chương trình dưới đây để tạo một danh sách liên kết trỏ bởi con trỏ head, gồm có hai phần tử node0 và node1:

```
ListNode *node0, *node1, *head;
  node0 = malloc(sizeof(ListNode));
  node1 = malloc(sizeof(ListNode));
  node0->num = 20;
  node1->num = 30;
  node0->next = ___ ;
  node1->next = ___ ;
  head = ___ ;
```

2) Hoàn thiện hàm printList()

3) Hoàn thiện hàm findNode()

```
ListNode *findNode(ListNode *head, int i) {
  ListNode *cur = head;
  if ((head == NULL) || (i < 0)) {
        printf("Danh sach lien ket rong hoac phan tu tim kiem khong ton tai");
return NULL;
  }
while (i > ___ ) {
        cur = ___ ;
        i = ___ ;
        if (cur == NULL) {
              printf("Phan tu tim kiem khong ton tai");
return NULL;
}
  }
  return cur;
}
```

4) Hoàn thiện hàm insertNode()

```
void insertNode(ListNode **pHead, int index, int value){
  ListNode *cur, *newNode;
  if (*pHead == ___ || index == ___ ){
            newNode = malloc(sizeof(ListNode));
            newNode->num = value;
            newNode->next = ___ ;
            *pHead = ___ ;
  }
  else if ((cur = findNode( ___, ___ )) != NULL){
            newNode = malloc(sizeof(ListNode));
            newNode->num = value;
            newNode->next = ___ ;
            cur->next =___ ;
} ese printf("can not insert the new item at index %d!\n", index);
}
```

5) Hoàn thiện hàm removeNode()

```
void removeNode(ListNode **ptrHead, int index){
  ListNode *cur, *pre;
  if (index == ___ ) {
            cur = ___ ;
            *ptrHead = ___;
            free(cur);
  } else {
            pre = findNode( ___ , ___);
            cur = ___ ;
            pre->next = ___ ;
            free(cur);
  }

}
```