Bài tập thực hành
Cấu trúc dữ liệu và thuật toán
Sắp xếp cơ bản (B)

1. Bài 1: Sắp xếp sản phẩm theo giá

Cho một danh sách các sản phẩm, mỗi sản phẩm có tên sản phẩm và giá sản phẩm. Viết chương trình bằng C để khai báo một struct san_pham để chứa các thông tin về sản phẩm, đọc dữ liệu về sản phẩm từ file, sắp xếp danh sách sản phẩm theo thứ tự giá tăng dần.

Dữ liệu vào trong file INPUT.TXT chứa mô tả cây nhị phân. Định dạng file INPUT.TXT như sau:

- Dòng đầu tiên chứa số nguyên N là số lượng sản phẩm.
- N dòng tiếp theo, mỗi dòng mô tả một sản phẩm theo định dạng: ten_san_pham gia_san_pham.
    - ten_san_pham: Chuỗi kí tự a-z và không chứa dấu cách.
    - gia_san_pham: Giá tiền của sản phẩm (đồng) có kiểu số nguyên dương.

Kết quả ghi ra file OUTPUT.TXT gồm N dòng, mỗi dòng ghi tên sản phẩm của dãy sau khi đã sắp xếp.

Ví dụ:

INPUT.TXT

```
5
laptop 10000000
mouse 120000
keyboard 100000
tablet 5000000
headphone 300000
```

OUTPUT.TXT

```
keyboard
mouse
headphone
tablet
laptop
```

2. Bài 2: Sắp xếp trên danh sách liên kết

Viết chương trình bằng ngôn ngữ lập trình C để hoàn thành bài tập sắp xếp tăng dần trên danh sách liên kết.

a) Khai báo cấu trúc dữ liệu một nút

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

b) Hoàn thiện hàm liệt kê các phần tử trong danh sách liên kết

```
void printList(ListNode *head) {
  ListNode *cur = head;
  while (cur != ___ ) {
        printf("%d", cur->num);
        if (cur->next != NULL)
              printf("->");
        cur = ___ ;
  }
}
```

c) Hoàn thiện hàm chèn một nút có giá trị data vào đầu danh sách liên kết trỏ bởi *phead:

```
void insertAtTheBegin(ListNode **phead, int data) {
  ListNode *ptr1 = malloc(sizeof(ListNode));
  ptr1->data = data;
  ptr1->next = ___ ;
  *phead = ptr1;
}
```

d) Hoàn thiện hàm sắp xếp danh sách liên kết dùng giải thuật sắp xếp lựa chọn

```
void selectionSort(ListNode *head) {
  ListNode *temp = head;
  while (temp) {
        ListNode *min = ___ ;
        ListNode *r = temp->next;
        while (r) {
              if (min->data > r->data)
                    min = ___ ;

              r = r->next;
        }
        int x = temp->data;
        temp->data = min->data;
        min->data = x;
        temp = ___ ;
  }
}
```

e) Hoàn thiện hàm sắp xếp danh sách liên kết dùng giải thuật sắp xếp nổi bọt

```
void bubbleSort(ListNode *head) {
```

```
  int swapped, i;
  ListNode *ptr1;
  ListNode *lptr = NULL;

  if (head == NULL)
          return;

  do {
          swapped = 0;
          ptr1 = head;

          while (ptr1->next != lptr) {
                  if (ptr1->data > ___ ) {
                          int temp = ptr1->data;
                          ptr1->data = ptr1->next->data;
                          ptr1->next->data = temp;

                          swapped = 1;
                  }
                  ptr1 =___ ;
          }
          lptr = ptr1; // vị trí đã sắp xếp đúng
  } while (swapped);
}
```

f) Hoàn thiện hàm sắp xếp danh sách liên kết dùng giải thuật sắp xếp chèn

```
void insertionSort(ListNode **phead)
{
  ListNode* sorted = NULL;
   ListNode* current = *phead;

   while (current != NULL) {
      ListNode* next = current->next;

      // Vi tri current se chen o dau danh sach sorted, hoac sorted == NULL
      if (sorted == NULL || sorted->data >= current->data) {
           current->next = sorted;
           sorted = current;
      }
      else {
         ListNode* t1 = sorted;

         // Xac dinh nut truoc vi tri can chen: current se chen sau t1
         while (t1->next != NULL
              && t1->next->data < ___ ) {
           t1 = ___ ;
```

```
            }
            current->next = ___ ;
            t1->next = current;
        }

        current = next;
    }

    *phead = sorted;
}
```

g) Viết hàm main() kiểm tra kết quả sắp xếp

```
int main(void) {
  int arr[] = { 10, 55, 33, 66, 99, 88 };
  int i;

  ListNode *head = NULL;

  for (i = 0; i < 6; i++)
            insertAtTheBegin(&head, arr[i]);

  printf("\n Linked list before sorting ");
  printList(head);

  //bubbleSort(head);
  //selectionSort(head);
  insertionSort(&head);

  printf("\n Linked list after sorting ");
  printList(head);

  return 0;
}
```