Bài tập thực hành
Cấu trúc dữ liệu và thuật toán
Cây tìm kiếm nhị phân (A)

1. Bài 1

1) Khai báo cấu trúc dữ liệu của một nút của cây tìm kiếm nhị phân như sau:

```
typedef struct _btnode {
  int item;
  struct _btnode *left;
  struct _btnode *right;
} BTNode;
```

2) Hoàn thiện hàm duyệt cây nhị phân theo thứ tự giữa:

```
void TreeTraversal_InOrder2(BTNode *cur) {
  if (cur == NULL)
        return;
  // Bổ sung code nếu cần
}
```

3) Hoàn thiện hàm trả về địa chỉ của nút có giá trị c trên cây tìm kiếm nhị phân có nút gốc là cur.

```
___  BSTT(BTNode2 *cur, int c) {
  if (cur == NULL)
        return NULL;
  if (c==cur->item)
        return ___;
  if (c < cur->item)
        return ___;
  else
        return ___;
}
```

4) Hoàn thiện hàm BSTT2(BTNode2 *cur, int c) thực hiện chèn một nút có giá trị c trên cây tìm kiếm nhị phân có nút gốc là cur. Hàm này trả về địa chỉ của nút mà nút có giá trị c sẽ là nút con khi chèn.

```
BTNode2* BSTT2(BTNode2 *cur, int c) {
```

```
  if (c==cur->item)
        return NULL;
  if (c < cur->item) {
        if (cur->left == ___ )
              return ___;
        return BSTT2(___);
  }
  else {
        if (cur->right == ___)
              return ___;
        return BSTT2(___);
  }
}
```

5) Hoàn thành hàm insertNodeBST() thực hiện chèn nút có giá trị c vào cây nhị phân tìm kiếm t, trả về con trỏ của nút giá trị c đã chèn vào cây:

```
BTNode* insertNodeBST(BTNode **t, int c) {
  if ((*t) == NULL) {
        (*t) = malloc(sizeof( ___ ));
        (*t)->item = c;
        (*t)->left = ___ ;
        (*t)->right = ___ ;
        return ___ ;
  }

  BTNode *posNode = BSTT2( ___ , ___ );

  if (posNode == NULL) {
        printf("Phan tu da ton tai");
        return NULL;
  }

  BTNode *btNewNode = malloc(sizeof(BTNode));
  btNewNode->item = c;
  btNewNode->left = NULL;
  btNewNode->right = NULL;

  if (c < posNode->item)
        ___  = btNewNode;
  else
```

```
            ___  = btNewNode;

  return btNewNode;
}
```

6) Hoàn thiện hàm tìm nút trái nhất trên cây tìm kiếm nhị phân t

```
BTNode* lastLeftChild(BTNode *t) {
  BTNode *p = t;
  while ( ___ != NULL) {
            p = ___ ;
  }
  return ___ ;
}
```

7) Hoàn thiện hàm xóa nút có giá trị num trên cây tìm kiếm nhị phân trỏ bởi *t

```
void removeBST(BTNode **t, int num) {
  if ( ___  > num) {
            removeBST(  ___ , num);
  } else if ( ___  < num) {
            removeBST( ___ , num);
  } else { // (*t)->item == num
            if (  (*t)->left == NULL) {
                      BTNode *r = *t;
                      (*t) = ___ ; // TH2/TH1
                      free(r);
            } else if ((*t)->right == NULL) {
                      BTNode *r = *t;
                      (*t) = ___ ; // TH2/TH1
                      free(r);
            } else { // TH3
                      BTNode *p = lastLeftChild( ___ );
                      (*t)->item = p->item;
                      removeBST(&((*t)->right), ___ );
            }
  }
}
```

8) Viết hàm main() chạy và giải thích kết quả:

```
int main() {
  char s[200] = "HEBACDFGLJIKM";
   BTNode *t = NULL;
  for (i = 0; i < n; i++) {
            insertNodeBST(&t, s[i]);
  }

  printf("\nKet qua duyet cay theo thu tu giua:\n");
```

```
  TreeTraversal_InOrder2(t);
  printf("\n");

  printf("\nKet qua duyet cay theo thu tu giua (sau khi xoa):\n");
  char c = 'H';
  removeBST(&t, c);
  TreeTraversal_InOrder2(t);

  printf("\nKet qua duyet cay theo thu tu giua (sau khi chen):\n");
  char c = 'Q';
  insertNodeBST(&t, c);
  TreeTraversal_InOrder2(t);

  return 0;
}
```