임베디드 애플리케이션 재시험

1. 한달간 C 언어 등등 많은 것들을 학습했는데 이것들에 대해 자기 성찰을 수행 해보자!

해당 과목에 대한 진심과 성의가 담긴 자기 성찰을 수행하도록 한다.
앞으로의 포부를 함께 기술하시오.
(배점 50 점)

이번 한달간 C언어를 기초부터 시작해서 이중포인터, 포인터 함수 등 높은 수준의 C언어 문법들까지 배웠는데 개인적으로 C언어에 자신이 있었는데도 모르는 부분이 생각보다 많이 있었고 아직 배워야 할 것이 매우 많다는 사실을 알게 되었습니다. 앞으로 어디를 가서도 C언어 잘한다는 소리 듣도록 열심히 배우겠습니다.

2. 프로그램을 작성하시오.

아무런 함수 3 개를 작성하여 프로그램이 동작하도록 만드시오.
(본인 만들고 싶은대로 만든다 - 함수 3 개 이상 쓰면 됨)

```
#include <stdio.h>
#include <string>
//              < pow()함수를 unsigned long형으로 변환 >
unsigned long pow_UL(unsigned long x, unsigned long y) {
        unsigned long result = 1;
        for (int i = 0; i < y; i++) {
                result *= x;
        }
        return result;
}
//               < 16진수 char -> 10진수 int 변환 함수 >
int HexCharToDec(char a) {
        if ('0' <= a && a <= '9') return a - '0';
        else if ('A' <= a && a <= 'F') return a - 'A' + 10;
        else if ('a' <= a && a <= 'f') return a - 'a' + 10;
        else return -1; //실패한 경우
}
//              < 16진수 str[] -> 10진수 int 변환 함수 >
int HexStrToDec(char *HexStr, int HexStr_len) {
        //HexStr[] -> MSB 순서
        int Dec = 0;
        for (int i = 0; i < HexStr_len; i++) {
                Dec += HexCharToDec(HexStr[HexStr_len - 1 - i])*pow_UL(16, i);
                if (Dec < 0) return -1; //실패한 경우
        }
        return Dec;
}
```

```
int main() {
        char str[] = "01fA";
        int result = HexStrToDec(str, 4);
        printf("result = %d₩n", result);
        return 0;
}
```

3. C 언어 메모리 레이아웃

C 언어 메모리 레이아웃을 기술하시오.

Text 영역 – 코드 그 자체가 기록되는 영역

Data 영역 – 전역변수나 정적인 자료가 저장되는 영역

Stack 영역 – 임시데이터(지역변수)들이 저장되는 영역

Heap 영역 – 동적할당 데이터들이 저장되는 영역

4. 함수 포인터는 왜 쓰는가 ?

알고 있는대로 기술하시오.

함수 포인터는 함수의 주소를 의미하는데

함수포인터를 사용하게 되면 함수의 인자에

상황에 따라 필요한 함수 포인터를 입력하여

원하는대로 실행시킬 수 있다.

5. 함수를 여러개 만드는 이유는 무엇인가 ?

서술형이니 기술하시오.

함수를 기능별로 분할하여 만들어두면 코드의 가독성이 좋아지고 함수의 기능이 필요한 상황에서 다시 코드를 만들 필요 없이 간단하게 함수만 불러와서 해결할 수 있다.

그리고 나중에 다른 프로젝트를 하게 되더라도 이전에 만들어두었던 함수가 필요하다면 다시 사용할 수 있어서 편리하다.

6. 포인터가 없는 언어들과의 차이점

포인터 때문에 할 수 있는 기능이 무엇인지 기술하시오.

다른 언어들과는 다르게 포인터를 이용하여 주소를 직접적으로 건드릴 수 있어서 위험하기도 하지만 사용자가 잘 사용한다면 함수 밖에 있는 변수들을 직접 변경하여 영향을 줄 수 있고 배열을 쉽게 다루는 등 메모리 공간을 효율적으로 제어할 수 있다.

자료구조 재시험

1. 이번 시험 결과와 관련하여 자료구조 과목에 대한 자기 성찰을 해보자!

자기 성찰과 앞으로의 포부를 기술하시오.
(자기 성찰을 자기 성찰 답게 대충적지 말고 진심과 성심을 다해 적도록 한다)

이번 자료구조 과목은 제가 충분히 익숙해질 정도로 연습을 하지 않아서 시간 내로 문제를 풀지 못했습니다. 그리고 AVL트리의 경우는 아직 이해도가 많이 부족하다는 생각이 듭니다. 앞으로 남는 시간에 틈틈이 자료구조와 재귀함수, 이중포인터 등에 익숙해지도록 필요한 부분을 꾸준히 연습하여 부족한 부분을 채워가겠습니다.

2. 연결리스트에 대한 문제
선입선출을 따르는 구조를 무엇이라 하는가 ?

큐(Queue)

3. 연결리스트 문제다.

후입선출 기능을 가진 자료구조를 무엇이라 하는가 ?

스택(Stack)

4. 트리에 관련한 개념 문제

트리 자료구조는 연결리스트에 비해 어떠한 이점을 가질 수 있는가 ?

Stack과 Queue는 선형 자료구조라서 원하는 자료를 검색하려면 최악의 경우 전체 자료의 갯수만큼 검색을 해야해서 속도가 매우 느린 반면 Tree의 경우는 자식노드 두개와 연결되어 있는 삼각형 모양의 비선형 자료구조라서 정해진 기준에 따라 자료들을 잘 분리해서 저장해놓을 경우 자료 검색 속도가 상대적으로 매우 빠르다는 장점이 있다.

5. AVL 트리 개념

AVL 트리와 일반 트리의 차이점이 무엇인가 ?

일반 트리의 경우 자료들이 들어가는 순서가 1, 2, 3, 4, 5 처럼 자료의 크기가 작은 순서 또는 큰 순서로 연속으로 들어가는 경우에는 비선형 자료구조의 이점을 살리지 못하고 자료들이 한쪽으로 치우쳐지게 되어 최악의 경우 선형 자료구조와 다를바 없는 상황이 일어날 수 있다.

반면 AVL트리의 경우 자료의 추가 삭제 과정에서 자료들이 한쪽으로 치우치는 현상을 막아주어서 비선형 자료구조의 이점을 최대한 살릴 수 있게 된다.

6. 레드 블랙 트리는 왜 사용하는가 ?

기존 트리들과 레드 블랙 트리의 차이점은 무엇인가 ?

대표적인 트리 중 하나인 AVL트리와 차이점을 비교해보면 AVL트리의 경우는 자료를 추가하거나 삭제할 때 마다 균형을 잡아주기 때문에 검색 속도는 Red-Black트리보다 빠르지만 균형을 잡아주는 시간 때문에 상대적으로 시간이 오래걸리는 단점이 있다.

반면 Red-Black트리의 경우는 매번 트리의 균형을 잡아주는 것이 아니라 자료 추가 또는 삭제과정 중 특정 조건이 성립되는 경우에만 균형을 잡아주기 때문에 자료의 추가 삭제 과정에서는 AVL트리보다 속도가 뛰어나다는 장점이 있다.