기말시험

[총 20점: 6 문항]
Q1, Q2: 각 2 점, **Q3, Q4**: 각 3 점, **Q5, Q6**: 각 5 점

컴프실 2022 (Spring)
cyclops@ajou.ac.kr

제출물

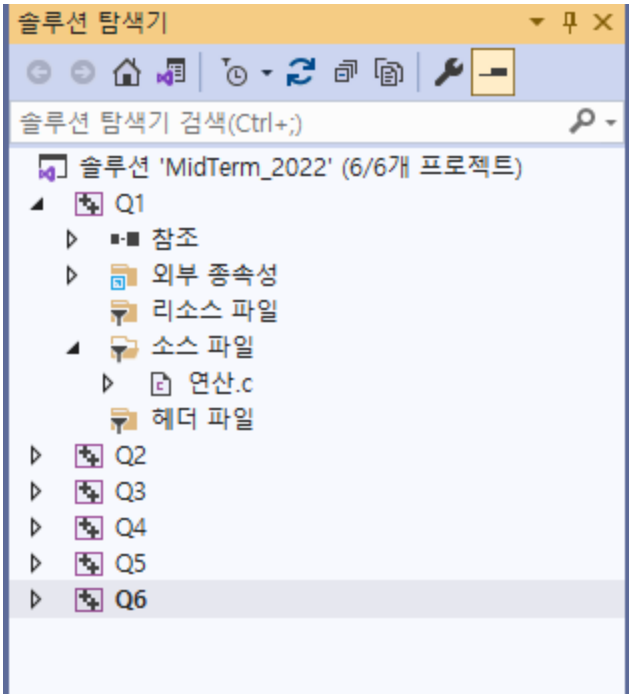
- 각 문항의 해답을 소스코드 포함하여 하나의 문서 파일로 (여러 개의 파일인 경우 하나의 압축 파일로) 제출합니다. 프로그램을 VS C로 구현한 경우, 아래 예와 같이, 각 문항의 프로그램이 개별 Project로 등록된 하나의 솔루션으로 구성하고, 이 솔루션 폴더를 압축한 파일을 제출하여도 됩니다..

※ 프로그램 작성시, 적절한 주석은 필수입니다. 특히, 설명을 요구하는 문항은 해답을 주석으로 작성합니다.

※ 실행되지 않는 코드는 0점 처리됩니다.

※ 각 문항의 해답을 같은 (동일) 솔루션 내의 개별 Project로 추가하여 구현할 수 있습니다. 각 문항 별 해는 main 함수를 가진 프로그램으로 작성되어, 선택적으로 실행시킬 수 있습니다. 이 경우도, MS VS의 **솔루션 폴더를 압축**하여 제출하면 됩니다. (솔루션을 생성할 때, 어떤 폴더를 사용하는지 확인하고, 제출 시 착오 없길 주의하세요.)

※ '.sln' 파일은 솔루션 관련 정보를 가진, VS가 관리하는 파일입니다. 이 파일만 제출하는 실수를 하지 마세요.



Q1. 아래 프로그램은, 변수 arr의 문자열 "GOOD"를, 포인터 P1과 P2를 사용하여, 각각 "FOOD"와 "WOOD"로 변경하여 출력하려고 만든 것입니다. 틀린 곳이 있으면, 그 이유를 설명하고, 올바르게 동작하도록 수정하세요. (2점)

```
void main() {
      char arr[20] = "GOOD";
      const char* p1;
      char* const p2;
      p1 = arr;
      p1[0] =  'F';
      printf("%s₩n", arr);
      p2 = arr;
      p2[0] = 'W';
      printf("%s₩n", arr);
}
```

Q2. 아래 함수 Count의 반환 값 N이 무엇을 나타내는지 살펴보고, 이 함수의 목적과 작동 원리를 설명하세요. (2점)

```
int Count(unsigned int x) {
  int N = 0;

  x ^= 0xFFFFFFFF
  while (x) {
    N++;
    x = x & (x - 1);
  }
  return N;
}
```

Q3. 아래 함수에 오류가 없는지 살펴보고, 오류가 있으면 정정을 하고, 정상적인 코드면, 이 함수의 목적과 작동원리를 설명하세요. (3점)

```
void duff(char *to,  char *from, int count)  {
    int n=(count+7)/8;
    switch (count%8) {
            case 0: do { *to++ = *from++;
            case 7:  *to++ = *from++;
            case 6: *to++ = *from++;
            case 5: *to++ = *from++;
            case 4: *to++ = *from++;
            case 3: *to++ = *from++;
            case 2: *to++ = *from++;
            case 1: *to++ = *from++;
            } while (--n >0);
    }
}
```

Q4. 아래의 매크로 함수, TripleXor가 어떤 목적으로 사용되는지 설명하고, 실수 변수에도 적용할 수 있도록, 아래 ①, ② 부분을 채워, 프로그램을 완성하세요. (3점)

```
#define TripleXor(a, b) ((a)^=(b)^=(a)^= (b))

int main() {
  float x = 4.19, y = 5.18;

  ①_____
  TripleXor(②_____) ;

  return 0;
}
```

Q5. 최소/최대 수 검색 (5점)

- 임의의 서로 다른 양의 정수 세 (3) 개를 입력 받아, 가장 작은 수와 가장 큰 수를 출력하는 C 프로그램을 작성합니다.

- 주의사항
  ① 3개의 서로 다른 양의 정수를 입력 받을 수 있는 친절한 메시지 출력하여, 입력 받는다.
  ② 입력된 세 수의 최소 수(Min)와 최대 수(Max)를 찾아 출력한다.
  ③ 입력 처리 후 결과 출력은 <입/출력 예시>를 따른다.
  ④ 입력된 첫 수가 음수이면 종료한다, 아니면, 위의 1,2항의 과정을 반복한다.
  ⑤ 프로그램 종료 시, 친절한 문구를 출력해줄 것.
  ⑥ 배열과 비교연산자는 사용하지 못한다. (사용하면 0 점 처리)

<입/출력 예시>

```
Enter Three Positive Integers: 3 6 9
Min (3, 6, 9) = 3
Max (3, 6, 9) = 9
Enter Three Positive Integers: 815 419 623
Min (815, 419, 623) = 419
Max (815, 419, 623) = 815
Enter Three Positive Integers: -1 0 0
Bye!!!
```

Q6. 문자열로 입력된 정수의 연산 (5점)

- 부호가 없는 정수 자료 형은, 32 bits으로 2^{32} (4,294,967,296)까지 나타낼 수 있고, 64 bits로는 2^{64} (18,446,744,073,709,551,615)까지 나타낼 수 있다. 이보다 큰 수들에 대한 연산 중 덧셈과 뺄셈을 할 수 있는 방법을 구현합니다. 두 (2) 개의 40자리까지의 10진수를 문자열로 입력 받아, 두 첫수에 둘째 수를 더하거나, 뺀 결과 값을 출력하는 프로그램을 작성합니다.
- 주의 사항
  - ① 프로그램 시작에 "첫째/둘째 수 (40자리 이하)"라는 메세지를 출력해 반복하여 입력을 받을 수 있도록 하고, 입력 받은 두수 모두 0이 입력되면 프로그램을 종료한다. (단. 아래의 <입력 함수>, getNumber를 호출하여 숫자 문자열을 입력 받는다.)
  - ② 입력 처리 후 결과 출력은 <입력/출력 처리 예시>를 따른다. 종료 시 친절한 문구를 출력해 줄 것. (다음 쪽 참조)
  - ※ 덧셈의 결과 값은 40자리를 초과할 수 있음.

<입력 함수: getNumber>

```
void getNumber(char s[]) {
  static int index = 0;
  static char n[][42] = {
    "12345678901234567890123456789012345678901234567890",
    "98765432109876543210987654321098765432109876543210",
    "1234567890",
    "9876543210",
    "12345678901234567890",
    "12345678901234567890",
    "12345678901234567890",
    "12345678901234567891"
  };
  if (index%2) printf("* 둘째 수 (40자리 이하): ");
  else printf("* 첫째 수 (40자리 이하): ");
  If (index < 8) {
    strcpy(s, n[index++]);
    printf("%s₩n", s);
  } else gets(s);
}
```

<입/출력 처리 예시>

```
* 첫째 수 (40자리 이하): 12345678901234567890123456789012345678901234567890
* 둘째 수 (40자리 이하): 98765432109876543210987654321098765432109876543210
  덧셈: 1111111110111111111101111111111011111111100
  뺄셈: -864197532086419753208641975320864197532086419753208641975320
* 첫째 수 (40자리 이하): 1234567890
* 둘째 수 (40자리 이하): 9876543210
  덧셈: 11111111100
  뺄셈: -8641975320
* 첫째 수 (40자리 이하): 12345678901234567890
* 둘째 수 (40자리 이하): 12345678901234567890
  덧셈:  24691357802469135780
  뺄셈:  0
* 첫째 수 (40자리 이하): 12345678901234567890
* 둘째 수 (40자리 이하): 12345678901234567891
  덧셈:  24691357802469135781
  뺄셈: -1
* 첫째 수 (40자리 이하): 1234
* 첫째 수 (40자리 이하): 1200
  덧셈:  2434
  뺄셈:  34
* 첫째 수 (40자리 이하): 0
* 첫째 수 (40자리 이하): 0
Bye!!!
```