# 01. int a[10][20]에서 배열이 차지하는 메모리 공간의 크기는 얼마인가? int형은 4바이트라고 하자.

(1) 200바이트 (2) 400바이트  
 (3) 600바이트 (4) 800바이트

배열 a는 총 200개의 정수형을 저장할 수 있다. 이때, 정수형은 각각 4바이트이므로 배열이 차지하는 메모리 공간의 크기는 800바이트이다.

# 02. float a[100]으로 선언된 배열의 시작 주소를 1000번지라고 할 때, 배열의 10번째 요소의 주소는 몇 번지인가?

(1) 1010번지 (2) 1020번지  
 (3) 1036번지 (4) 1040번지

float형이 각각 4바이트이므로 배열의 10번째 요소는 1000 + (9 \* 4)번지, 즉 1036번지이다.

# 03. 다음 배열 중에서 크기가 가장 큰 배열은?

(1) int array1[10]; (2) double array2[10];  
 (3) char array3[40]; (4) float array4[10];

배열들 array1[], array2[], array3[], array4[]은 각각 40바이트 (4\*10), 80바이트 (8\*10), 40바이트 (1\*40), 그리고 40바이트 (4\*10)이므로 크기가 가장 큰 배열은 array2이다.

# 04. 크기가 10인 배열 two[]를 선언하고 여기에 2의 제곱 값들을 저장해보자. 즉 배열의 첫번째 요소에는 2⁰을 저장하고 두번째 요소에는 2¹값을 저장한다. 마지막 요소에는 2⁹값을 저장한다. for 루프를 이용하여 two[] 배열의 전체 요소의 값을 출력하는 프로그램을 작성하라.

#include <stdio.h>  
  
int power(int n) {  
 if (n == 0) return 1;  
 return (2 \* power(n-1));  
}  
  
int main() {  
 int two[10];  
  
 for (int i = 0; i < 10; i++) {  
 two[i] = power(i);  
 printf(“%d\t” two[i]);  
 }  
 return 0;  
}

# 05. person이라는 구조체를 만들어보자. 이 구조체에는 문자 배열로 된 이름, 사람의 나이를 나타내는 정수 값, 각 개인의 월급을 나타내는 float 값 등이 변수로 들어가야 한다.

struct person {  
 char name[10];  
 int age;  
 float salary;  
}

# 06. C언어의 typedef을 이용하여 complex라고 하는 새로운 자료형을 정의하라. complex 자료형은 구조체로서 float형인 real 변수와 역시 float형인 imaginary 변수를 갖는다. complex 자료형으로 변수 c1과 c2를 선언하여 보라.

typedef struct {  
 float real, imaginary;  
} complex;  
  
int main() {  
 complex c1, c2;  
 return 0;  
}

# 07. 복소수를 구조체로 표현해보자. 복소수 a와 복소수 b를 받아서 a+b를 계산하는 함수를 작성해보자. 함수는 구조체를 반환할 수 있다. 알다시피 복소수는 real + imag\**i*와 같은 형태를 갖는다.

typedef struct {  
 float real, imaginary;  
} Complex;  
  
Complex complex\_add(Complex a, Complex b) {  
 Complex new;  
 new.real = a.real + b.real;  
 new.imaginary = a.imaginary + b.imaginary;  
 return new;  
}

# 08. 크기가 n인 배열 array에서 임의의 위치 loc에 정수 value를 삽입하는 함수 insert()를 작성하라. 정수가 삽입되면 그 뒤에 있는 정수들은 한 칸씩 뒤로 밀려야 한다. 현재 배열에 들어있는 원소의 개수는 items개라고 한다. (여기서 items << n라고 가정)

void insert(int array[], int loc, int value) {  
 if ((loc >= n) || (loc < 0) {  
 exit;  
 } else if ((loc >= items) && (loc < n)) {  
 array[loc] = value;  
 items += 1;  
 } else if ((loc >= 0) && (loc < items)) {  
 for (int i = (items-1); i >= loc; i--) {  
 array[i+1] = array[i];  
 }  
 array[loc] = value;  
 items += 1;  
 }  
}

# 09. 앞의 문제에서 구현한 insert() 함수의 시간 복잡도는?

O(n)

loc = 0이고 items = (n-1)일 때 모든 요소들을 한 칸씩 뒤로 밀려야 되는 시나리오가 된다. 이 경우, for 루프를 n-1번 들어가고 마지막에 array[0]에 value 값을 넣으므로 O(n)의 시간 복잡도를 가지게 된다.

# 10. 크기가 n인 배열 array에서 임의의 위치 loc에 있는 정수를 삭제하는 함수 delete()를 작성하라. 정수가 삭제되면 그 뒤에 있는 정수들은 한 칸씩 앞으로 이동하여야 한다. 현재 배열에 들어있는 원소의 개수는 items개라고 하자. (여기서 items << n라고 가정)

int delete(int array[], int loc) {  
 if ((loc >= n) || (loc < 0) {  
 exit;  
 } else if ((loc >= items) && (loc < n)) {  
 array[loc] = 0;  
 items -= 1;  
 } else if ((loc >= 0) && (loc < items)) {  
 for (int i = loc; i < (items-1); i++) {  
 array[i] = array[i+1];  
 }  
 array[items-1] = 0;  
 items -= 1;  
 }  
 return 0;  
}

# 11. 앞의 문제에서 구현한 delete() 함수의 시간 복잡도는?

O(n)

loc = 0이고 items = n일 때 모든 요소들을 한 칸씩 앞으로 이동하여야 하는 시나리오가 된다. 이 경우, for 루프를 n-1번 들어가고 마지막에 array[items-1]에 0 값을 넣으므로 (요소를 없애는 역할) O(n)의 시간 복잡도를 가지게 된다.

# 12. 1개의 정수와 최대 크기가 20인 문자열로 이루어진 구조체를 저장할 수 있도록 동적 메모리를 할당받고 여기에 정수 100과 문자열 “just testing”을 저장한 다음, 동적 메모리를 반납하는 프로그램을 작성하라.

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
  
typedef struct {  
 int integer;  
 char character[20];  
} myStructure;  
  
int main() {  
 myStructure \*p;  
  
 p = (myStructure \*)malloc(sizeof(myStructure));  
 if (p == NULL) {  
 fprintf(stderr, ”메모리가 부족해서 할당할 수 없습니다.\n”);  
 exit(1);  
 }  
 p->integer = 100;  
 strcpy(p->character, “just testing”);  
 free(p);  
  
 return 0;  
}