단원 16 모의고사 점수:( ) 등급:( )

일자 : 20 년 월 일 학과 _____ 학번 _____ 이름 _____

1. 다음에서 서술내용이 맞으면 O, 틀리면 X 하시오. (10 = 5문항 * @2)

① 변수 선언에 의한 정적 방식은 사용이 간편하나 실행 이전에 사용할 메모리의 공간 크기를 모두 알아야 한다. (O)

② 함수 malloc()은 인자로 메모리 할당의 크기를 지정하고, 반환 값으로 할당된 메모리의 시작 주소를 반환한다. (O)

③ 동적 메모리 할당 함수는 malloc(), calloc(), realloc() 3가지이다. 동적으로 할당된 메모리를 해제하는 함수는 close()이다. (X)

④ 연결 리스트는 배열에 비하여 임의 접근(random access)도 빠른 장점이 있다. (X)

⑤ 사용자정의 헤더파일도 시스템 헤더파일과 같이 함수원형, 매크로, 자료형 재정의에 관련된 문장으로 구성된다. (O)

2. 다음에서 빈 부분을 적절히 채우시오. (20 = 5문항 * @4)

① 동적 메모리 할당(dynamic memory allocation)은 실행 중에 메모리를 할당 방식으로 (힙)이라는 메모리 영역에 할당된다.

② 함수 malloc()은 메모리 공간을 확보하고 초기 값을 저장하지 않는다. 함수 (calloc())은 메모리 공간을 확보하고 초기 값을 자료형에 알맞게 0을 저장한다.

③ 연결 리스트는 첨자대신 (링크(link))라는 포인터로 다음 노드를 가리키는 구조이다.

④ 지시자 #include에서 (큰 따옴표)를 사용해서 사용자정의 헤더파일을 삽입한다.

⑤ 연산자는 (##)은 매크로 정의 지시자 #define에서 사용되는 인자를 다른 토큰들과 연결해주는 연산자이다.

3. 다음 각각의 문제에서 물음에 알맞은 것을 고르시오. (20 = 5문항 * @4)

① 다음은 메모리 할당에 대한 설명이다. 다음 중 잘못된 것은 무엇인가? (다)
가) 정적 메모리 할당 방식에서는 프로그램이 실행되기 이전에 변수의 저장 공간 크기를 정해야 한다.
나) 프로그램 실행 중에 필요한 메모리를 할당하는 방법이 동적 메모리 할당이다.
다) 동적 메모리로 할당된 메모리는 close() 함수호 메모리 해제한다.
라) 동적 메모리 할당 방식은 함수를 이용한다.

② 다음은 메모리 할당에 관련된 문장이다. 다음 중 잘못된 문장은 무엇인가? (나)

가) int *pi = (int *) malloc(sizeof(int));
나) int *pc = (int *) calloc(sizeof(int)*3);
다) double *p= (double*)malloc(sizeof(double));
라) int *pr = (int *) realloc(NULL, sizeof(int));

③ 동적 메모리가 할당되는 메모리 영역은 다음 중 무슨 영역인가? (다)
가) data
나) stack
다) heap
라) all

④ 함수 malloc()이 실패한 경우 반환 값은 다음 중 무엇인가? (다)
가) 0이 아닌 값
나) 1
다) NULL
라) -1

⑤ 다음 중 전처리 연산자가 아닌 것은 무엇인가? (라)
가) ##
나) #@
다) defined
라) #undef

4. 다음 프로그램에서 출력 결과를 기술하시오. (10)

```
#include <stdio.h>
#include <stdlib.h>
double * append(double *, double *, int, int);

int main(void)
{
        int i;
        double *p1 = (double *) calloc(5, sizeof(double));
        double *p2 = (double *) calloc(3, sizeof(double));
        p2[0] = 1.4; p2[1] = 2.3; p2[2] = 3.6;
        p1 = append(p1, p2, 5, 3);

        for (i = 0; i < 8; i++)
                printf("%.1f ", p1[i]);
        puts("");
        return 0;
}

double * append(double *a, double *b, int x, int y)
{
        int i;
        double *p=(double *)realloc(a, (x+y) *
sizeof(double));
        for (i = 0; i < 3; i++) {
                p[x+i] = b[i];
                printf("%.1f ", p[x+i]);
        }
        puts("");
        return p;
}
```

5. 다음 프로그램의 빈 네모상자 부분에 들어갈 소스를 작성하시오. (40)

- 다음 구현에서 함수 search()는 연결 리스트 head에서 멤버 data가 d인 노드를 찾아 반환하는 함수

```
#include <stdio.h>
[                                        ]

struct node {
        double data;
        struct node *link;
};
[                                        ]

node * createNode(double);
int printList(node *);
node * search(node *, double);
```

```
int main(void)
{
        node *head = NULL, *tail = NULL;
        node *cur = NULL;

        head = createNode(0.5);
        tail = head;
        for (int i = 1; i<5; i++)
        {
                tail->link = createNode(i + 0.5);
                tail = tail->link;
        }
        printList(head); puts("");
        puts("검색 시작: ");
        cur = search(head, 3.5);
        printf("검색한 노드는 %.1f₩n", cur->data);

        return 0;
}

//연결리스트 head에서 노드의 data가 d인 노드를 찾아
반환하는 함수
node * search(node *head, double d)
{
[                                        ]
}

//노드를 생성하는 함수
node * createNode(double n)
{
        //구현을 가정
}
```

이 모의고사 성적에 대한 등급 의견입니다.

등급	점수	의견
최상위	90 이상	매우 우수한 수준입니다.
상위	80 이상	우수한 수준입니다
보통	70 이상	보통 입니다.
하위	60 이상	노력이 필요합니다.
최하위	60 미만	많은 노력이 필요합니다.

<수고 하셨습니다.>

4.

5.

```
#include <stdio.h>
#include <stdlib.h>

struct node {
   double data;
   struct node *link;
};
typedef struct node node;

node * createNode(double);
int printList(node *);
node * search(node *, double);

int main(void)
{
        int i;
        node *head = NULL, *tail = NULL;
        node *cur = NULL;

        head = createNode(0.5);
        tail = head;
        for (i=1; i<5;i++)
        {
                tail->link = createNode(i+0.5);
                tail = tail->link;
        }
        printList(head);
        cur = search(head, 3.5);
        printf("%.1f₩n", cur->data);

        return 0;
}

//연결리스트head에서노드의data가d인노드를찾아반환하
는함수
node * search(node *head, double d)
{
        int cnt = 0;
        node *nextNode = head;
        while (nextNode == NULL || nextNode->data !=
d)
        {
                printf("%d 번째노드는%.1f₩n", ++cnt,
nextNode->data);
                nextNode = nextNode->link;
        }
        return nextNode;
}

//노드를생성하는함수
node * createNode(double n)
{
        node *cur;
        cur = (node *) malloc(sizeof(node));
        if (cur == NULL)
        {

        printf("노드생성을위한메모리할당에문제가있
습니다.₩n");
                return NULL;
        }
        cur->data = n;
        cur->link = NULL;

        return cur;
}


//연결리스트의모든노드출력함수
int printList(node *head)
{
        int cnt = 0;
        node *nextNode = head;
        while (nextNode != NULL)
        {
                printf("%d 번째노드는%.1f₩n", ++cnt,
nextNode->data);
                nextNode = nextNode->link;
        }

        return cnt;
}
```