Name: Hemos ID:

Student ID (학번):

CSED-101 PROGRAMMING & PROBLEM SOLVING

Spring 2014

Midterm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Q12 | Total |
| Your Score |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Max  Score | 5 | 18 | 7 | 4 | 6 | 5 | 7 | 7 | 10 | 8 | 16 | 7 | 100 |

* Write down your name, hemos ID, and student ID.
* There are 14 pages in this midterm.
* Your answers must run correctly in **C programming language** without error or warning. Otherwise, your answers will be considered incorrect. For example, it is ok to put more parentheses than needed in your answer, but it will be incorrect if you put fewer parentheses than needed.
* You must write your answer on the underline => \_\_\_\_\_\_\_\_. Scratches outside the underline will be ignored.
* The total score is 100
* This is a 3-hour exam.

|  |
| --- |
| **명예서약서**  **나는 명예로운 포스테키안으로서 다음의 Honor Code를 지킬 것을 맹세합니다.**  **정직과 타인에 대한 존중이 함께하는 포스테키안의 미래는 명예롭다.**  **2014년 월 일**  **이름 (인)** |

1. (5점) 아래 condition의 결과가 true 이면 T, false 이면 F 로 답하시오.

단, 각 문항 별 1점 배점, 오답일 경우 -1점, 공백의 경우 0점.

int a = 1, b = 2, c = 3, d = 4, e = 5, x = -9, y = -3, z = 0;

* 1. [ \_\_F\_\_\_\_\_\_\_\_ ] a < c > b
  2. [ \_\_T\_\_\_\_\_\_\_ ] 4 \* d / 5 % 3 && d
  3. [ \_\_T\_\_\_\_\_\_\_\_ ] d == (e++) – 1
  4. [ \_\_T\_\_\_\_\_\_\_\_ ] y == (a += b\*= c -=5)
  5. [ \_\_F\_\_\_\_\_\_\_\_ ] z || x < (y – 5) && !y

해설

이 문제는 연산자 우선순위에 대한 것으로 연산자 우선순위 표를 참고하면서 보게 되면 더 쉽게 이해하실 수 있습니다.

* 1. a< c > b에서 대소 관계비교연산자는 결합방향이 <-입니다. 그렇기 때문에 c >b가 먼저 수행되고 c>b는 true이므로 1이 되고, 그 다음의 비교는 a<1이 됩니다. 그렇기 때문에, 1<1이 되고, 이것은 false가 됩니다.
  2. 여기서 우선순위가 가장 높은 것은 \* / % 이고 결합방향은 ->입니다. 그러므로 4\*d 는 16이고, 16/5=3 (int 이므로) 3%3=1 그 다음 1 && d 에서 d는 0이 아닌 상수 이므로 true가 됩니다.
  3. 후위 연산자 및 괄호가 가장 우선순위이므로, e++가 수행 되지만 ++는 모든 연산이 끝난 뒤 수행되므로 고려하지 않아도 됩니다. 그러므로 e-1=4 이고, d==4는 참이므로 true입니다.
  4. 괄호 내부에 있는 두 연산을 합친 += -= 등은 결합방향이 <-이고 우선순위가 같습니다. 그리고 괄호 안에 있으므로 먼저 수행됩니다. 그러면 이 때, c-=5에서 c=-2가 되고, 이어서 b\*=-2이므로 a+=-4가 되어서 a=-3이되고, 이것을 y==-3 으로 비교하게 되면 true입니다.
  5. 괄호 내부의 y-5를 가장 먼저 계산하면 -8이고 그 다음 우선 순위인 x<-8을 계산하면 true입니다. 그 다음은 ||인데 true가 이미 존재하므로 z값에 관계 없이 true이고 true && !y에서 y는 true이므로 !y=false 입니다. 그러므로 값은 false 가 됩니다. (!y는 사실상 y-5가 수행된 뒤 바로 수행됩니다.)

1. (18점) 다음 문제를 읽고 답하시오.
   1. (1점) 다음 중 변수명으로 사용할 수 없는 것은? \_\_\_\_\_\_\_\_\_\_

① cs101 ②$option ③ A4 ④ IF ⑤ \_2dNum

해설

변수명으로 사용할 수 있는 것은 숫자, 영어 대문자, 소문자, \_(언더바) 입니다. 그러나 변수의 맨 처음 앞 글자로 올 수 있는 것은 영어 대문자,영어 소문자, 언더바 입니다. 그러므로 특수문자가 들어있는 2번은 변수명이 될 수 없습니다.

* 1. (1점) 다음 중 함수 선언(declaration)이 올바르게 된 것은? \_\_\_\_\_\_\_\_\_\_

① int fun(int &x, int y);

② int fun(int x, y);

③ void fun(float x, int y);

④ void fun(x, y);

해설

함수의 선언은 [리턴 타입] [함수의 이름] ( 인자들 ) 로 구성 됩니다. 이 때, 인자들을 선언할 때에는 (int x, int y) 와 같이 각각의 변수이름 앞에 모두 데이터 타입이 나와야 합니다. 이러한 정보들을 토대로 문제를 살펴보면, 정답이 될 수 있는 것은 3번 뿐이라는 것을 알 수 있습니다.

* 1. (3점) x=1, y=2, z=3 인 경우, 다음 코드를 실행 시킨 후의 x, y, z 의 값은?

|  |  |
| --- | --- |
| switch ( z > y )  {  case 0: x--;  y--;  case 1: x--;  z--;  default: y--;  z--;  } | x : \_\_\_\_\_0\_\_\_\_\_  y : \_\_\_1\_\_\_\_\_\_\_  z : ­­\_\_\_\_\_1\_\_\_\_\_ |

해설

Switch(z>y)에서 z>y는 true이므로 1이되고, 그러므로 case 1 로 흐름이 흘러 갑니다. 그런데, 여기서 함정이 되는 부분이 있는데, 바로 이 구문에 break; 가 존재하지 않아서 case 1 이후로 모든 명령이 다 실행됩니다. 그러므로 x--,z--,y--,z—가 실행 될 것이고, 그러면 z는 2만큼 줄어들고 y와 x는 각각 1만큼 줄어들 것입니다.

그러므로 x : 0, y : 1, z : 1 이 됩니다.

* 1. (2점) x=10, y=40 인 경우, 다음 코드를 실행 시킨 후의 x 의 값은?

|  |  |
| --- | --- |
| do  {  x += 9;  }while(x < y); | x : \_\_\_\_46\_\_\_\_\_\_ |

해설

이 프로그램을 잘 분석해 보면 x는 9씩 더해지고, x>=y가 될 때의 즉, 반복문을 탈출할 때의 x값이 몇이 될지를 구하는 문제 입니다. 그러면 10부터 9의 배수만큼 더하여 40보다 커지는 첫 번째 수는 10 + 9 \* 4 = 46 일 때입니다. 즉, x=46이 됩니다.

* 1. (3점) x=0, y=1 인 경우, 다음 코드를 실행 시킨 후의 x, y 의 값은?

|  |  |
| --- | --- |
| for(x = 10; x > 0; x = x – 3)  {  y = y \* x;  } | x : \_\_\_\_-2\_\_\_\_\_\_  y : \_\_\_\_\_280\_\_\_\_\_ |

해설

이 프로그램은 x가 10부터 3씩 줄어들며, y에 곱해지는 반복문의 형태를 띠고 있습니다. 그리고 x<=0 일 경우에 이 반복문을 빠져나가게 됩니다. 그러면 y에 곱해지는 값은 10, 7, 4, 1이 될 것입니다. 그러므로 y의 값은 10\*7\*4\*1 = 280이 될 것이고, 그렇다면 x의 값은 반복문을 빠져 나가며, 1-3=-2가 될 것입니다.

* 1. (2점) x=0, y=0 인 경우, 다음 코드를 실행 시킨 후의 x, y 의 값은?

|  |  |
| --- | --- |
| for(x = 0; x < 10; x++)  {  if ( x % 2 == 0 )  continue;  y += x;  } | x : \_\_\_\_10\_\_\_\_\_\_  y : \_\_\_\_\_25\_\_\_\_\_ |

해설

이 반복문은 x가 0부터 1씩 증가하고, x<10일동안 반복문이 진행됩니다. 그런데 if문을 보면 2로 나눈 나머지가 0일때, 즉 짝수 일 때 아무런 코드도 실행시키지 않고 continue;문을 사용하는 것을 볼 수 있습니다. Continue; 문은 그 아래에 있는 코드를 아무것도 실행하지 않고, 바로 다음의 흐름으로 가져갑니다. 예를 들어 x가 2이면 짝수이므로 아무것도 실행하지 않고 바로 x는 3이됩니다. 그러므로 x의 값은 루프를 빠져 나오므로 10이 될 것이고, y는 1~9까지의 모든 홀수의 합인 1+3+5+7+9 = 25가 될 것입니다.

* 1. (4점) x=0, y=0, z=0 인 경우, 다음 코드를 실행 시킨 후의 x, y, z 의 값은?

|  |  |
| --- | --- |
| for (x = 0; x < 10; x += 3)  for (y = x + 1; y < 10; y += 5)  z += x + y; | x : \_\_\_12\_\_\_\_\_\_\_  y : \_\_\_10\_\_\_\_\_\_\_  z : \_\_\_39\_\_\_\_\_\_\_ |

해설

x=0에서부터 3씩 증가하고, x<10일동안 루프가 돌게 됩니다. 그리고 그 안에서 두 번째 루프로 y=x+1부터 y<10 일 동안 5씩 증가하면서 루프가 돌고 있습니다. 그리고 그 두 번째 루프마다 z에는 x+y만큼의 값이 더해지고, 그러면 루프를 차근차근 쫓아가면 x=0일 때, y=1,6으로 두 번째 루프를 수행하게 됩니다. 그러므로 x=0일 때, z=0+1 + 0+6 =7이 됩니다. 그 다음, x가 3으로 증가하고, y=4, 9이 올 수 있게 됩니다. 그러면, z= 7 + ( 3+4) +(3+9) = 26가 됩니다. 그 다음은 x가 6이 되고, y=7이 됩니다. 이 때, z =26 + 6+ 7 = 39가 됩니다. 그런 뒤 x가 9가 되면 y는 10이 되어서 두 번째 루프는 수행되지 못합니다. 그러므로 x는 바로 한 바퀴 더 돌아서 12인 상태로 나가고, y는 이전에 10으로 초기화 되었으므로 y = 10, z=39가 됩니다.

* 1. (2점) x=10, y=10, z=0 인 경우, 다음 코드를 실행 시킨 후의 z 의 값은?

|  |  |
| --- | --- |
| while (x--)  do  z = x + y;  while (--x); | z : \_\_\_\_\_\_\_\_\_\_ |

해설

이 함수에서 z에 수행되고 있는 것은 어차피 대입문이므로 가장 마지막에 z에 대입되는 값만을 찾게 되면, z의 값을 알 수 있습니다. 그러면 이 z=x+y가 가장 마지막으로 실행되는 것은 x--가 0이 될 때, 즉 x가 1일 때 마지막으로 수행 된다는 것을 알 수 있습니다. 왜냐하면 do~while 구문에서는 뒤에 있는 while의 조건을 검사하기 때문입니다. 그러면 x가 1일 때, z= 1 + 10이 수행되며, 반복문은 종료되고, 이어서 맨 처음 while(x--)도 x가 0이었으므로 종료 됩니다. 그 이후에 x값을 출력 하라는 문장이 있으면, -1이 될 것입니다. 그런데 여기서는 z값만 원하므로 답은 11이 됩니다.

1. (7점) 아래의 프로그램을 실행하였을 때, 화면에 출력되는 결과를 쓰시오.

|  |
| --- |
| #include <stdio.h>  #define ADD 6 + 3  int f1(int a, int b);  int f2(int a, int b, int c);  int f3(int a, int b);  void swap(int a, int b);  int main(void)  { int a = 2, b = 4, c = 5;  printf(“%d %d\n”, a, f1(a, b)); //(1)  c += f2(a, b, c);  printf(“%d\n”, c); //(2)  printf(“%d\n”, f3(c, 0)); //(3)  swap(a, b); //(4)  printf(“%d %d\n”, a, b); //(5)  printf(“%d\n”, ADD / 3); //(6)  return 0;  }  int f1(int a, int b)  {  b = a + b;  a = 10;  return b;  }  int f2(int a, int b, int c)  {  --a; --b; --c;  return –a;  }  int f3(int a, int b)  {  for (a = -4; a < 4; a++)  if (a + 1) b += a;  return b;  }  void swap(int a, int b)  {  int temp = a;  a = b;  b = temp;  printf(“%d %d\n”, a, b);  } |

출력 결과

|  |  |
| --- | --- |
| (1) (1점) | 2 6 |
| (2) (1점) | 4 |
| (3) (2점) | -3 |
| (4) (1점) | 4 2 |
| (5) (1점) | 2 4 |
| (6) (1점) | 7 |

해설

이 문제는 사용자 정의함수에 대한 이해와 call-by-value함수를

잘 이해하고 있는 가에 대한 문제입니다.

1. a와 f1(a,b)를 출력하고 있는 문장입니다. 그러면 f1을 분석하여 보자. F1은 b에 a+b를 저장하고, b를 리턴하는 함수입니다. 이 때 , a = 10;이라는 구문은 신경 쓸 필요가 없습니다. 왜냐하면, call-by-value함수이기 때문에 a값은 변하지 않습니다. 그러므로 출력값은 2와 6 이 될 것입니다.
2. 2번에서는 f2로 계산한 값을 c에다 더한 값을 출력하는 것입니다. F2(a,b,c)에서는 요점적으로 a--구문을 통해 a에서 1을 빼고, -a를 리턴하는 함수입니다. (1)과 같이 그 외의 값의 변화는 무시해도 됩니다. 그러면 c+=-1이 되어 c=4가 되고, 그러므로 (2)의 답은 4가 됩니다.
3. 여기서는 f3(c,0)의 값을 물어보고 있습니다. F3을 보면, a+1이 참일 동안에 a값을 계속 b에 더합니다. 그러면 a+1이 거짓이 되는 값은 a+1이 0, 0.0 \0일 때이므로, a=-1일때만, 이 조건문은 거짓이 됩니다. 그러면 b에 더해지는 값은 -4+(-3)+(-2)+0+1+2+3이 되고, a=4가 되면서 반복문을 빠져나옵니다. (물론 메인의 a는 변하지 않습니다.) 그리고 리턴값은 b이므로 정답은 -3이 됩니다.
4. swap함수 내부에 있는 printf구문에서 출력 되는 값을 알아내는 문제입니다. 여기서 a값과 b값은 실제로 한번도 변한 적이 없으므로, swap함수에 들어가는 두 개의 숫자는 2 와 4일 것입니다. 함수 내부에서 swap이 이루어 진 뒤, 함수 내부에서 값을 출력했으므로 값은 서로 뒤바뀐 상태 그대로 일 것이므로 답은 4 2 가 됩니다.
5. 메인함수 내에서 a와 b를 출력하는 간단한 문제입니다. 그러나 앞에서 a와 b값이 변경되었다고 생각하면 큰일이 납니다. 지금까지의 모든 함수는 call-by-value함수이었고, 그 속에서 일어난 값의 변화는 전혀 영향을 미치지 못하므로 출력값은 2 4가 됩니다.’
6. define전처리기 지시자는 ADD 라는 것을 말 그대로 6+3으로 치환해버립니다. 그대로 치환해 버리기 때문에 ADD / 3은 6+3/3으로 치환이 되는데, 여기서 연산자 우선순위 때문에 나눗셈이 먼저 수행되고 그러면 6+3/3 = 6+1 = 7이 됩니다.
7. (4점) 아래 프로그램은 사용자로부터 두 숫자와 하나의 문자를 순서대로 입력 받아, 사칙 연산을 계산하는 프로그램입니다. 이 프로그램이 정상적으로 작동할 수 있도록 빈 칸을 채우시오.

|  |
| --- |
| #include <stdio.h>  int main(void)  {  float a, b;  char c;  while (1)  {  scanf(“%f %f”, &a, &b);  scanf(“ %c”, &c);  if (\_\_\_\_\_\_\_c==’+’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)  {  printf(“%f + %f = %f\n”,\_a\_\_\_\_,\_\_\_b\_\_,\_\_a+b\_\_\_);  }  else if (\_\_c==’-‘\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)  {  printf(“%f - %f = %f\n”,\_\_a\_\_\_,\_b\_\_\_\_,\_a-b\_\_\_\_);  }  else if (\_\_\_c==’\*’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)  {  printf(“%f \* %f = %f\n”,\_a\_\_\_\_,\_\_b\_\_\_,\_\_a\*b\_\_\_);  }  else if (\_\_c==’/’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)  {  printf(“%f / %f = %f\n”,\_a\_\_\_\_,\_\_b\_\_\_,\_\_a/b\_\_\_);  }  else  {  printf(“End\n”);  break;  }  }  return 0;  } |

해설

이 프로그램에서 무슨 연산을 할 지 결정해주는 것은 문자 c입니다. 그러면 이 문자 c가 조건문에 들어가 c의 값에 따른 분기를 두어야 할 것입니다. printf문 내부에서의 출력 값들은 각각 연산에 알맞은 값으로 정의 해주면 될 것입니다. 그 때의 코드는 위의 코드와 같습니다.

1. (6점) 아래의 프로그램은 4개의 정수를 입력 받아, 덧셈 및 뺄셈을 수행한 후 그 결과를 출력하는 프로그램입니다. 아래의 실행 결과를 참조하여 빈 칸을 채우시오. 단, +, -, \*, /, %와 같은 연산자를 사용하지 않고, 오직 **add** 와 **sub** 함수만을 이용해서 작성합니다.

**(실행예시)**

|  |
| --- |
| Insert four integer values: 1 2 3 4  1+2+3+4=10  1-2-3-4=-8 |

|  |
| --- |
| #include <stdio.h>  int add(int x, int y);  int sub(int x, int y);  int main()  {  int a, b, c, d, e, f;  printf(“Insert four integer values: “);  scanf(“%d %d %d %d”, &a, &b, &c, &d);    e = \_\_\_\_\_\_\_\_add(a,add(b,add(c,d)))\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;    f = \_\_\_\_\_\_\_sub(sub(sub(a,b),c),d)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  printf(“%d+%d+%d+%d=%d\n”, a, b, c, d, e);  printf(“%d-%d-%d-%d=%d\n”, a, b, c, d, f);  return 0;  }  int add(int x, int y)  {  return x + y;  }  int sub(int x, int y)  {  return x – y;  } |

해설

연산자가 없이 두 개의 정수연산을 수행하는 함수를 이용해서 네 개의 정수에 대해서 수행해야 합니다. 그러면 a+b+c+d는 a+(b+(c+d))라는 결과를 거치면 두 정수의 덧셈 3번으로 축약 될 수 있을 것입니다. 또한 sub에서는 (((a-b)-c)-d)와 같은 결과를 통해서 계산할 수 있을 것입니다. 그러면 각각 답은 위와 같이 나타납니다. 물론 이 외에도 몇 가지 답이 더 존재합니다. 그것들에 대해서도 한번 시도해보기 바랍니다.

1. (5점) 아래 프로그램은 주사위 2개를 굴려서 나온 숫자의 합이 짝수면 사용자가 이기고, 홀수면 지는 프로그램입니다. 주사위에서 나올 수 있는 숫자의 범위는 1~6입니다. 프로그램이 정상 작동하도록 빈 칸을 채우시오.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  \_int randDice(void)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void printResult(int a);  int main(void)  {  int dice1, dice2;  srand(time(NULL));  dice1 = randDice();  dice2 = randDice();  printf(“Dice 1: %d, Dice 2: %d\n”, dice1, dice2);  \_printResult(dice1+dice2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  return 0;  }  \_\_\_int randDice(void)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  {  return \_\_\_rand()%6+1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  void printResult(int a)  {  if (\_a%2\_\_\_\_)  printf(“The sum of the 2 dice is %d. You lose.\n”, a);  else  printf(“The sum of the 2 dice is %d. You win.\n”, a);  } |

해설

처음 해야 할 것은 함수의 프로토 타입 선언입니다. 함수의 프로토 타입 선언은 함수의 원형을 선언하는 방식으로 문제 2.2에서 보았던 형식과 같아야 합니다. 메인에는 있지만 함수에는 없는 함수를 살펴보면 randDice()가 존재하고, 인자는 하나도 없고, 그 리턴값은 int형 변수에 저장 한다는 것을 통해서 int randDice(void)가 함수의 원형임을 알 수 있습니다. 그리고 이것은 주사위의 눈 중 하나를 랜덤하게 리턴 해야 하므로 return rand()%6+1이 됨을 알 수 있습니다. 6으로 나눈 나머지는 0~5까지 이므로 1을 더하면, 1~6까지의 숫자를 임의로 리턴 할 수 있습니다. 그러고 나면 printResult함수를 완성해야 하는데 문제에서 보듯이 홀수일 때 게임에서 패배함을 알 수 있는데, 조건문을 만족하면 패배하므로 조건은 홀수가 되는 조건이 들어와야 할 것입니다. 이 조건을 만족시키려면 나머지 연산을 이용하면 되는데 즉, a%2가 될 것입니다. a%2가 애초에 0 또는 1이 연산의 결과이기 때문에 비교연산자를 사용할 필요가 없습니다. 그리고 이렇게 홀짝 판별을 하는 것은 두 주사위 눈의 합이므로 함수 호출시에는 printResult(dice1+dice2)로 호출해야 할 것입니다.

1. (7점) 아래 프로그램은 3자리 양의 정수를 랜덤하게 생성하려는 프로그램입니다. 단, 각 자리의 숫자는 서로 중복되는 경우가 없다는 조건을 만족시켜야 합니다. 빈 칸을 채우시오.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #define TRUE 1  #define FALSE 0  int is\_duplicate(int n);  int main()  {  int num;  srand(time(NULL));  do  {  num = \_\_\_rand()%900+100\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }while(\_\_\_\_\_\_\_is\_duplicate(num)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);  printf(“%d\n”, num);  return 0;  }  int is\_duplicate(int n)  {  int n100, n10, n1;  int check = FALSE;  n1 = \_\_\_\_\_\_\_n%10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  n10 = \_\_\_\_\_\_\_\_n/10%10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  n100 = \_\_\_\_\_\_\_\_\_n100%10\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  if(\_\_\_n1==n10 || n1==n100 || n10==n100 \_\_\_\_\_\_\_\_\_\_\_\_\_)  check = TRUE;  return \_\_\_\_\_\_\_\_\_;  } |

해설

이 프로그램에서 num은 세 자리의 정수를 임의로 받아야 하고, is\_duplicate함수는 자릿수에 중복되는 숫자가 있는지 검사하는 역할을 하게 될 것입니다. 그러므로 num=100~999까지의 값이 들어와야 하므로 num=rand()%900+100이 되고, while내부의 조건은 is\_duplicate(num)이 됩니다. 그러면 is\_duplicate를 구성해보면, n1,n10,n100에는 각각의 자릿수가 나타나야 합니다. 이런 자릿수는 나머지 연산과 나누기 연산으로 얻어낼 수 있는데, n1 = n % 10 이 될 것이고, n10 = n/10 % 10 이 되며, n100 = n/100 % 10 이 될 것입니다. 이렇게 정수 나눗셈의 버림과 나머지를 이용하여, 자릿수를 얻어내고 나서 if문의 조건에서는 겹칠 경우 true가 되어 반복되어야 하기 때문에, 겹치는 조건문을 만들어야 합니다. 그러므로,

n1==n10 || n1==n100 || n10==n10 이 적절한 답이 될 것입니다. 그리고 마지막 return에서는 check를 리턴해야 true 또는 false가 리턴되면서 반복을 제어할 수 있을 것입니다.

1. (7점) 피보나치 수는 아래의 점화식으로 정의되는 수열입니다.
   1. (4점) 피보나치 수를 ***int fibo (int n)*** 이라는 함수를 사용하여 구현하고자 합니다. 이 함수를 **recursion**을 이용하여 작성하시오.

|  |
| --- |
| #include <stdio.h>  int fibo(int n);  int main(void)  {  int seriesSize = 12;  int looper;  printf(“First %d Fibonacci numbers:”, seriesSize);  for (looper = 0; looper < seriesSize; looper++)  {  if(looper % 4)  printf(“, %4d”, fibo(looper));  else  printf(“\n%4d”, fibo(looper));  }  printf(“\n”);  return 0;  }  int fibo(int n)  {  if (\_\_\_\_\_n==0 || n==1\_\_\_\_\_\_\_\_\_\_\_\_\_)  return \_\_\_\_\_\_\_\_\_n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  else  return \_\_\_\_\_fibo(n-1) + fibo(n-2)\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  } |

해설

피보나치 수열을 위에 정의된 점화식을 통해 구해야 합니다. Recursion을 사용해야 하고, n=0일때 0, n=1일 때 1이므로 만약 n==0이거나 n==1이면 각각에 맞는 n값을 리턴하고 나머지 수에서는 fibo(n-1)+fibo(n-2)를 리턴해야 할 것입니다. 그러므로 코드는 위의 모습과 같이 됩니다.

* 1. (3점) 위의 작성된 프로그램을 실행하였을 때, 화면에 출력되는 결과를 쓰시오.

|  |
| --- |
| First 12 Fibonacci numbers:  0, 1, 1, 2  3, 5, 8, 13  21, 34, 55, 89 |

해설

이 출력값은 %4d와 반복문 속의 조건문이 looper %4인 것으로 보아 4번의 주기로 한 줄씩 띄워지며 그 출력 숫자들은 4칸의 공백을 두고 출력 되어야 합니다. 그러한 과정을 거쳐서 피보나치 숫자들을 하나씩 출력하여 보면 위와 같은 출력결과가 나타납니다. (글자를 출력하는 printf문을 빼먹지 않도록 합니다.)

1. (10점) 다음 문제를 읽고 답하시오.
   1. (5점) 아래 프로그램은 사용자로부터 양의 정수 1개를 입력 받아, 숫자의 순서를 거꾸로 출력하는 프로그램입니다. 프로그램이 정상 작동하도록 빈 칸을 채우시오.

|  |
| --- |
| #include <stdio.h>  void reverseInt(int n);  int main(void) {  int n;  scanf(“%d”, &n);  reverseInt(n);  return 0;  }  // 아래의 함수는 recursion 을 이용하여 완성하시오. (재귀 함수)  void reverseInt(int n)  {  if(\_\_n<10\_\_\_\_\_\_\_) // (1점)  printf(“%d”, n);  else  {  \_\_\_\_printf(“%d”,n%10)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_reverseInt(n/10)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  } |
| **[실행 결과]**  **입력: 54321**  **출력: 12345** |

7번 문제에서와 같은 아이디어를 이용하면, 쉽게 풀어 낼 수 있습니다. 1의 자릿수부터 차근차근 출력하게 된다면, 역순으로 출력할 수 있을 것입니다. 역순의 값을 한번에 얻어낼 수 있는 곳을 살펴보면 바로 n<10일 때입니다. N>10이라면 n을 10으로 나눈 나머지를 출력하고, n을 차근차근 10으로 나누며 자릿수를 출력하면 될 것입니다. 그렇다면 코드는 위와 같이 구성됩니다.

* 1. (5점) 위의 함수 **reverseInt를 while문**을 이용하여 완성하시오. (재귀 함수로 작성할 수 없음)

|  |
| --- |
| void reverseInt(int n)  {  while(\_\_\_n>0\_\_\_\_\_\_\_)  {  \_\_\_\_\_printf(“%d”,n%10)\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_\_n/=10\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  } |

해설

이 함수를 while문을 이용해서 구현한다고 하더라도 같은 아이디어를 이용합니다. 자릿수를 차근차근 출력 하면 되는데 조건문을 나가는 부분을 생각해보면, 모든 자릿수가 빠져 더 이상 값이 나오지 않을 때 즉 n이 0보다 작거나 같아지는 구간에서 반복문이 종료 될 것입니다. 9.1과 같은 맥락으로 코드를 구성하면 위와 같습니다.

1. (8점) 다음은 주어진 배열의 순서를 뒤집는 프로그램입니다. 배열의 각 원소는 처음에는 배열에서의 자신의 위치 값을 할당합니다. 프로그램이 정상적으로 작동하도록 빈칸을 채우시오. 추가적인 변수는 선언할 수 없으며 ARY\_SIZE를 최소 6번 사용해야 합니다.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define ARY\_SIZE 10  int main()  {  \_int arr[i];\_\_\_\_\_\_\_\_\_\_\_\_\_  int i;  int temp;  // 초기 값을 지정해준다.  for (i = 0; i < \_ARY\_SIZE\_\_\_\_\_\_\_\_\_\_\_ ; i++)  {  \_arr[i]=I;\_\_\_\_\_\_\_\_\_\_\_\_\_  printf(“%d “, \_\_\_arr[i]\_\_\_\_\_\_\_\_);  }  printf(“\n”);  // 배열의 순서를 뒤집어 저장합니다.  for (i = 0; i < \_\_ARY\_SIZE/2\_\_\_\_\_\_\_\_\_\_ ; i++)  {  \_temp=arr[i]\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_arr[i]=arr[ARY\_SIZE]\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_arr[ARY\_SIZE]=temp\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  // 바뀐 값을 출력합니다.  for (i = 0; i < \_\_ARY\_SIZE\_\_\_\_\_\_ ; i++)  {  printf(“%d “, \_\_arr[i]\_\_\_\_\_);  }  printf(“\n”);  return 0;  } |
| **실행결과:**  0 1 2 3 4 5 6 7 8 9  9 8 7 6 5 4 3 2 1 0 |

해설

처음 밑줄은 배열을 선언해야 합니다. 배열을 선언하는 방법은 [데이터 타입] [배열이름][[원소의 개수]]로 이루어집니다. 그러므로 int arr[ARY\_SIZE]; 로 선언해주면 됩니다. (세미콜론을 주의합니다.) 그 다음 배열의 초기값은 인덱스 값으로 지정하라고 되어있으므로 ARY\_SIZE까지 돌며, arr[i]에 들어갈 값은 i가 되면 됩니다. 즉 arr[i]=I; 그리고, 역순으로 출력하는 부분이 관건인데, 변수 선언이 불가능 하므로 i와 ARY\_SIZE-i의 값을 서로 바꾸면, 중앙을 기준으로 양쪽이 바뀌므로 역순으로 배열 되게 될 것입니다. 단지, 이 때는 루프는 ARY\_SIZE/2까지만 돌아야 할 것입니다. 그렇지 않으면 다시 원래 순서로 배열될 것이기 때문입니다. 그런식으로 구성하게 되면 코드는 다음과 같이 구성됩니다.

1. (16점) 아래의 프로그램은 파일(input.txt)로부터 학생들의 시험 성적을 읽고 각 학생의 총점(평균으로 계산)과 학점 및 과목평균을 계산하여 화면에 출력하는 프로그램입니다. 아래 빈칸을 채워 프로그램을 완성하시오. 단, 2개의 변수만 선언해서 사용할 수 있습니다.

파일에는 줄(line) 단위로 한 학생의 학번, 중간고사 점수, 기말고사 점수가 순서대로 기록되어 있습니다.

**(input.txt)** **(출력화면)**

|  |
| --- |
| 201301 89 78  201302 76 84  201303 67 42  201304 66 88  201305 98 99 |

|  |
| --- |
| **================**  **학번 총점(학점)**  **================**  **201301 83.5 (B)**  **201302 80.0 (B)**  **201303 54.5 (F)**  **201304 77.0 (C)**  **201305 98.5 (A)**  **================**  **과목평균: 78.70**  **================** |

|  |
| --- |
| #include <stdio.h>  float calc\_average(int x, int y);  char calc\_grade(float score);  int main()  {  FILE \*infile;  int stu\_id, mid, final;  float average;  char grade;  \_int num\_of\_stu=0;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_float tot\_ave=0;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  infile = fopen(\_\_”input.txt”\_\_\_\_\_\_\_\_, \_\_\_\_”r”\_\_\_\_\_\_\_);  if (infile \_==NULL\_\_\_\_\_\_\_\_\_\_)  {  printf(“cannot open input file.\n”);  return 1;  }  printf(“=================\n”);  printf(“학번\t총점(학점)\n”);  printf(“=================\n”);  while( fscanf(\_infile, ”%d%d%d”\_\_, \_\_&stu\_id,&mid,&final\_) \_!=EOF\_)  { // (5점) 아래의 5줄 중 4줄만 채우면 됨. 아래에서 변수 선언 할 수 없음  \_\_\_\_num\_of\_stu++\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_average=calc\_average(mid,final)\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_grade=calc\_grade(average)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_tot\_ave+=average\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  printf(“%d\t%.1f (%c)\n”, stu\_id, average, grade);  }  printf(“=================\n”);  printf(“과목평균: %.2f\n”, tot\_ave / num\_of\_stu );  printf(“=================\n”);  fclose(infile);  return 0;  } // main  // 함수의 입력 값으로 전달 받은 x와 y의 평균을 계산  float calc\_average(int x, int y)  {  return \_\_\_(float)(x+y)/2\_\_\_\_\_\_\_\_\_  }  // 학점 계산  char calc\_grade(float score)  {  char grade;  switch ( \_\_\_score/10\_\_\_\_\_\_\_\_\_\_\_\_ )  {  case 10:  case 9 : grade = ‘A’;  break;  case 8 : grade = ‘B’;  break;  case 7 : grade = ‘C’;  break;  case 6 : grade = ‘D’;  break;  default: grade = ‘F’;  } //switch  return grade;  } |

해설

파일 입출력에 관한 문제입니다. Fopen에 관해서 설명하자면, fprint를 할 때는 파일에 쓰기 위한 용도이므로 “w” 쓰기모드로 받아야 하고, fscanf 파일에서 데이터를 가져오고 싶다면, “r”읽기모드로 fopen을 해주어야 합니다. 여기서는 fscanf 를 사용하고 있기 때문에 읽기 모드로 열어주어야 할 것입니다. 그러므로 infile=fopen=(“input.txt”,”r”);이 됩니다. 그리고 이 파일을 열지 못할 경우에는 NULL을 반환하기 때문에 그 다음 조건문은 infile ==NULL일 때입니다. 그 다음 fscanf는 파일포인터, 포맷스트링, 변수 주소 와 같은 순서로 받는다. 또한 더 이상 받을 수 없을 때 EOF를 리턴하게 됩니다. EOF가 되면 받는 작업을 그만해야 하므로 fscanf(infile , “%d%d%d”,&stu\_id,&mid,&final)!=EOF 가 됩니다. 그리고 그 내부에서는 학생의 숫자를 더하고, 총 평균의 합과 학생 각자의 평균을 계산하는 작업을 수행하여야 합니다. 그리고 이 때 학생의 숫자와 총 평균의 합을 저장할 변수 두 개를 선언 해야 하므로, 변수선언의 답은 int num\_of\_stu=0; 과 float tot\_ave=0;입니다. 그리고 calc\_grade함수에서는 switch문을 통해 학점을 얻어내기 위하여, 점수를 10으로 나누어 10의 자리부분만 떼어내어 검사하고 있으므로, switch(score/10)이 올 것입니다. 그렇게 코드를 구성하면 위와 같이 됩니다.

또한 calc\_average에서는 casting을 꼭 빼먹지 않는 것이 중요합니다. 두 정수의 연산이기 때문에 casting은 필수적입니다.

1. (7점) 아래의 프로그램은 배열의 원소 중 가장 큰 값을 찾아 그 값을 출력하는 프로그램입니다. 빈 칸을 채우시오.

배열 list 는 아래의 그램과 같이 정수형의 원소로 구성되어 있습니다.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| list | 0 | 5 | 7 | 3 | 2 | 6 | 9 | 1 | 4 | 8 |

|  |
| --- |
| #include <stdio.h>  #define ARY\_SIZE 10  int get\_max(int arr[], int size);  int main()  {  // (2점) 배열 list 를 위의 그림과 같이 초기화 하시오.  \_int list[ARY\_SIZE]={0,5,7,3,2,6,9,1,4,8}\_\_\_\_\_\_\_\_  int max\_value;  \_max\_value = get\_max(list,ARY\_SIZE);\_\_ // (2점)함수 get\_max() 호출  printf(“MAX = %d\n”, max\_value);  return 0;  } // main  int get\_max(int arr[], int size)  {  int i;  int max = \_arr[0]\_\_;  for (i = 1; i < size; i++)  {  if( \_\_max<arr[i]\_\_\_\_ )  {  \_\_\_\_max=arr[i];\_  }  } //for  return max;  } // get\_max |
| **실행 결과:**  MAX = 9 |

해설

여기서 가장 큰 값을 searching할 때의 요점은 배열의 맨 처음 값을 저장해 놓은 뒤, 그것보다 큰 값이 나타나면 그 값으로 교체를 하면서 배열의 맨 끝까지 탐색하고 나면, 가장 큰 값이 max라는 변수에 들어있을 것이라는 알고리즘으로 진행됩니다. 또, 배열의 원소 초기화는 선언과 동시에 중괄호를 통해서 이루어집니다. 그리고 배열의 이름은 주소이므로, call-by-reference이고, 그러므로 다른 함수 내에서도 값의 수정이 가능합니다. 이런 정보들을 가지고 코드를 구성하면 위와 같습니다.