프로그래밍 언어 중간 시험

2017.4.19

1. 추상화 (abstraction)이라는 것은 어떤 사물을 나타내는데 있어서 대표적인 특징만을 이용하여 나타내는 것을 의미한다. 다음의 물음에 답하라. (9점)
   (a) 프로그래밍 상에서 이런 추상화를 하는 이유는 무엇인가 ? (3점)
   (b) 대부분의 프로그래밍 언어에서 제공하는 대표적인 추상화 방법 2가지를 설명하고, 각 추상화 방법에서 어떤 부분을 어떻게 추상화하였는지 예를 들어 설명하라. (각 3점 x 2 =6점)

2. 대부분의 언어에서는 변수의 type을 프로그램에서 미리 선언(declaration)함으로서 compiler가 이런 선언을 통하여 코드를 생성할 시 필요한 그 변수에 대한 많은 정보를 얻을 수 있다. compiler가 이런 type 선언을 통하여 얻을 수 있는 정보와 이런 정보를 이용하여 하는 일들에 대하여 설명하라. (예를 들면 type checking을 수행하여 type-error를 찾는 것도 이런 일들 중의 하나이다.) (10점)

3. 다음과 같은 C 언어 프로그램에 대하여 다음의 물음에 답하여라. (21점)

```
#include <stdio.h>
int aa[5] = {11,22,33,44,55};

main() {
  int bb, *cc ;
  cc = &(aa[1]);
  printf("%d\n", (int) cc);
  cc += 2 ;
  bb = (*cc)++ ;
}
```

   (a) 이 프로그램 수행이 끝나고 난 후에 aa[]의 내용은 무엇인가 ? (3점)
   (b) 프로그램이 끝나고 난 후에 bb에 저장된 내용은 무엇인가 ? (3점)
   (c) 이 프로그램에서 printf()에 의하여 프린트된 cc의 내용이 1000이라고 가정하면, 수행이 끝나고 난 후에 cc에 저장된 내용은 무엇인가 ? (3점)
   (d) 이 프로그램을 gcc로 컴파일 하면 타입에 대한 warning error가 나온다. 어느 statement가 warning error인지 그 이유와 함께 설명하라. (3점) (함수 prototyping에 대한 error는 제외하고 설명하여라.)
   (e) C언어에서 pointer의 type을 하나로 제한하는 이유는 무엇인가 ? (3점)
   (f) <stdio.h> file에 저장되어 있는 printf()에 대한 정보는 다음과 같다.

```
extern int printf (const char *__restrict __format, ...);
```

   (가) 일반적으로 이런 header file에 저장된 정보 및 용도를 설명하라. (3점)
   (나) 이 정보를 통하여 알 수 있는 printf() 함수의 parameter 형식을 설명하라. (3점)

4. 프로그래밍 언어에서의 binding은 여러 싯점에서 일어난다.
다음과 같은 binding이 일어나는 싯점은 각각 언제인가 ? (각 3점 x 8 = 24점)
(a) heap dynamic array의 index range binding
(b) union 변수의 storage size binding
(c) limited dynamic length string 변수의 storage binding
(d) static scoping rule을 사용하는 언어에서 어떤 변수의 scope binding
(e) 프로그램상의 '&' 심볼에 대한 동작 (operation) 바인딩
(f) enumeration 타입 변수의 storage 크기 바인딩

5. 다음 Python 프로그램의 출력 결과를 써라. (5점 x 3 = 15점)

(a) (5점)

```
for num in range(10,15):
   for i in range(2,num):
      if num%i == 0:
         j=num/i
         print '%d equals %d * %d' % (num,i,j)
         break
   else:
      print num, 'is a prime number'
```

range()함수 수행 예제

```
>>> # One parameter
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
>>> # Two parameters
>>> for i in range(3, 6):
...     print(i)
...
3
4
5
>>> # Three parameters
>>> for i in range(4, 10, 2):
...     print(i)
...
4
6
8
```

(b) (5점)

```
edibles = ["ham", "spam","eggs","nuts"]
for food in edibles:
    if food == "spam":
        print("No more spam please!")
        break
    print("Great, delicious " + food)
else:
    print("I am so glad: No spam!")
print("Finally, I finished stuffing myself")
```

(c) (5점)

```
edibles = ["ham", "spam", "eggs","nuts"]
for food in edibles:
    if food == "spam":
        print("No more spam please!")
        continue
    print("Great, delicious " + food)
    # here can be the code for enjoying our food :-)
else:
    print("I am so glad: No spam!")
print("Finally, I finished stuffing myself")
```

6. 아래 두개의 C 프로그램의 수행 결과를 각각 설명하고, dl 예제를 이용하여 C 언어의 operand evaluation order에 대한 규칙을 설명하라. (7점)

(a)

```
#include <stdio.h>
int sub();
int  k, j ;
void main() {
  k = 5  ;
  j = k  + sub() ;
  printf("k=%d, j=%d\n", k, j) ;
}
int sub() {
k = 3;
return(k);
}
```

(b)

```
#include <stdio.h>
int sub();
int  k, j ;
void main() {
  k = 5  ;
  j = k  + sub() ;
  printf("k=%d, j=%d\n", k, j) ;
}
int sub() {
k = 3;
return(k);
}
```

7. 아래 프로그램의 출력 결과 ((a) ~ (e))를 써라. (단, word alignment를 한다고 가정한다. print되는 인수에 주의 할 것.)(21점)

```
#include <stdio.h>

char aaa ;
enum days {Mon, Tue, Wed, Thu};


char bbb[2][3]={'a', '\0'};

union utag {
          char bb; float kk ; char *c ;
} haha ;

struct aa {
  float j; char aaa; enum days bb; char bbb;  union utag kk;
} lala, test[6][10] ;

void main() {
 printf("Size of char pointer : %d\n", sizeof(char *));
 printf("Size of integer : %d\n", sizeof(int));
 printf("Size of float : %d\n", sizeof(float));
 printf("%d \n", &(test[0][0]));
 printf("%d %d \n", sizeof(bbb), strlen(bbb));          /* (a): 3점 */
 printf("%d \n", sizeof(haha));                         /* (b): 3점 */
 printf("%d \n", sizeof(test));                         /* (c): 3점 */
 printf("%d %d \n",&(test[3][4].kk.kk),
                   &(test[3][4].kk.kk)+1);              /*(d) : 6점 */
 printf("%d %d \n",&(test[4][11]),  &(test[4][11])+1); /* (e) : 6점 */
}
```

앞 4개의 print 문장 출력 결과

```
Size of char pointer : 8
Size of integer : 4
Size of float : 4
5744
```

프로그래밍 언어 중간 시험

2018.4.26

1. 다음의 용어들을 간단히 설명하라. (각 3점 x 10 = 30점)
   (a) Exception Handling (b) Discriminated Union (c) Lifetime and Scope
   (d) History Sensitive Variable (e) Orthogonality
   (f) Dynamic Type Binding의 장단점 (g) Loop Parameters

2. 아래 같은 C 언어 프로그램은 컴파일시 error가 발생한다. 그 이유를 설명하라. (5점)

```
main() {
  int x ;
  &x = (int *) malloc(sizeof (int)) ;
}
```

3. 아래와 같은 C 프로그램에 대하여 다음의 물음에 답하라. Multi byte 저장시 Big Edian으로 저장된다고 가정한다. (각 5점 x 2 = 10점)

```
main() {
  int i = -1 ;          /* (a) */
  float j = 0.5 ;       /* (b) */
}
```

   (a) 변수 i는 4 바이트로 저장된다. i가 저장된 메모리 상의 4바이트를 Hex로 써라.(5점)
   (b) 변수 j는 4 바이트로 저장된다. j가 저장된 메모리 상의 4바이트를 Hex로 써라.(5점)

4. 다음과 같은 C-like한 언어 프로그램이 Intel 80386 (32bit 처리기)을 사용하는 컴퓨터에서 수행된다고 가정하고 다음의 물음에 답하라. (단, word-alignment를 하며, set 변수를 저장할 시 꼭 필요한 byte 수만을 할당하며, float 및 double은 IEEE 754 표준으로 저장되며, array는 row-major형태로 저장된다고 가정한다.) (답안을 작성할 시 유도하는 식도 꼭 같이 쓸 것) (20점
   (a) lala 변수를 저장시키는데 필요한 메모리 양은 몇 byte인가 ? (10점)
   (b) test[][]의 메모리 상에서의 시작 주소가 1000번지라고 가정하였을 때, "&(test[2][3].u_tag) + 1"가 지칭하는 주소는 어디인가 ? Array index는 0부터 시작한다고 가정한다.(10점)

```
type colors = (a,b,c,d,e,f,g,h,i);
type colorset = set of colors ;
type index = 'a'..'z';
union utag {
   char b[2] ;
   char *c ;
   index kk ;
} ;
struct {
  index i ;
  colorset mycolor ;
  float j ;
  char  c ;
  union utag aa ;
} test[10][10], lala ;
```

5. 오른쪽과 같은 프로그램은 compile시에 error가 발생한다. (15점)
   (a) 어떤 이유로 compile error가 발생하는지 설명하라. (5점)
   (b) error가 발생하지 않도록 수정하라. (5점)
   (c) 수정하고 난 뒤에 수행시키면 run-time error가 발생하는가 ? 발생 여부를 이유와 함께 설명하라. (5점)

```
#include <stdio.h>
main() {
  int a[100][100] ;
  sub1(a) ;
}
sub1(x)
int x[][] ; {
  x[98][100] = 5 ;
}
```

6. (Imperative Language)
   (a) 기존의 imperative 프로그래밍 언어 기본 설계에 영향을 미친 요소들에 대하여 설명하고, 그들이 프로그래밍 언어에는 어떤 형태로 나타났는지 설명하라. (6점)
   (b) 어떤 같은 문제를 해결하기 위하여 C와 FORTRAN과 같은 Imperative Language로 작성한 프로그램과 PROLOG나 LISP과 같은 declarative 언어로 작성한 2개의 프로그램이 있다고 가정하자. 이 두 프로그램을 LINUX가 설치된 Pentium PC에서 똑 같이 컴파일한 후에 수행시켰을 때 일반적으로 어떤 언어로 작성한 프로그램의 수행 속도가 빠른가 ? 그 이유와 함께 설명하라. (4점)

7. 프로그래밍 언어에서의 binding은 여러 시점에서 일어난다. 다음과 같은 binding이 일어나는 시점은 각각 언제인가 ? (각 3점 x 7 = 21점)
   (a) static local 변수의 storage 바인딩
   (b) C 언어에서 포인터 변수의 타입 바인딩
   (c) 함수 코드에 대한 주소 바인딩
   (d) union 변수의 storage 크기 바인딩
   (e) C 언어에서 formal parameter에 대한 value 바인딩
   (f) malloc()에 의하여 할당된 변수의 타입 바인딩
   (g) Set 변수에 대한 storage size 바인딩

8. 다음과 같은 Pascal-like한 프로그램에서 아래와 같은 5가지 parameter-passing 방법들에 의하여 parameter가 전송된다고 가정하였을 때 다음 프로그램의 output은 무엇인가 ?
   단, by result 및 by copy인 경우에 actual parameter의 주소는 <u>값이 return될 때 계산된다고</u> 가정하고, 최종 값을 알 수 없는 경우는 "unknown"으로 답할 것. (3점 x 5 = 15점)

```
procedure BIGSUB
  integer GLOBAL;
  integer array LIST[1:2];
  procedure SUB(PARAM);
    integer PARAM;
    begin
      PARAM := 3;
      GLOBAL := GLOBAL - 1;
      PARAM := 5 ;
    end
  begin /* Main */
    LIST[1] := 3;
    LIST[2] := 1;
    GLOBAL := 2;
    SUB (LIST[GLOBAL]);
    print(GLOBAL, LIST[1], LIST[2]);
  end
```

   (a) by value
   (b) by result
   (c) by copy
   (d) by reference
   (e) by name

프로그래밍 언어 중간 시험

2019.4.25

1. 다음의 용어들을 간단히 설명하라. (각 3점 x 4= 12점)
   (a) CISC와 RISC에서 HLL 언어 프로그램 수행을 빠르게 하기 위하여 사용한 방법들에 대한 설명
   (b) Exception handling을 표현할 수 있는 언어를 이용하면 신뢰도 있는 프로그램을 쉽게 작성(easy to program)할 수 있는 이유
   (c) User-defined abstract data type
   (d) Procedure-oriented Programming과 Data-oriented Programming 방법론에서는 대규모 프로그램 생성시 다른 방법을 사용한다. 공통점과 차이점을 설명하라.

2. 아래의 C 프로그램을 이용하여 다음의 물음에 답하라. (11점)

```
#include <stdio.h>
int two[2][] = {1,2,3,4};
main() {
    char ch = 'C';  /* 'C'의 ASCII code는 67이다. */
    int i; float f;
    f = i = ch + 1;
    printf("%c ", i); /* (b) */
    printf("%d, %f\n",i, f); /* (c) */
}
```

   (a) 위의 프로그램을 컴파일하면 error가 난다. Compile error가 나는 곳을 지적하고, 그 이유를 설명하라. (5점)
   (b)-(c) error가 없도록 수정한 후 이 프로그램에 의하여 프린트 되는 내용을 써라. (각 3점 x 2 = 6점)

3. Dangling Pointer 및 Garbage에 대하여 다음 빈 칸을 채워라. (각 2점x10 = 20점)

	Dangling Pointer	Garbage
(a) 정의	①	②
(b) 생성되는 이유(예를 들어 설명)	③	④
(c) 프로그램 수행 상의 문제점	⑤	⑥
(d) 해결 방법 (각 2가지)	⑦	⑨
	⑧	⑩

4. 대부분의 프로그래밍 언어에서 정수 (integer)의 표현은 일반적으로 수행되는 컴퓨터 CPU의 word size로 저장한다. 32 bit CPU상에서 수행되는 C 프로그램을 가정하고 아래 물음에 답하라. (20점)

(a) 표현할 수 있는 정수 값의 최소 값과 최대 값을 Hexadecimal (16진수)로 표현하라. 단, 음수는 2's complementary 형식으로 저장된다고 가정한다. (5점)

(b) 다음 프로그램을 Intel 86계열 32-bit CPU상에서 수행되고, 변수 i의 시작 번지가 1000번지부터 저장된다고 가정한다.

```
int main(){
    int i = 0x00000001;
    if( ((char *)&i)[0] ){
        printf( "Little Endian\n" );}
    else {
        printf( "Big Endian\n" );
    }
}
```

(가) 메모리상의 i 변수가 저장된 1000번지 ~ 1003번지의 값을 순서대로 Hexa로 써라 (5점)

(나) 위의 프로그램의 출력 값은 무엇인가? 그 이유와 함께 설명하라. (5점)

(c) Java 언어로 작성된 아래 프로그램을 16-bit CPU와 32-bit CPU에서 수행되는 Web 브라우져상에서 수행시키면 각각 어떤 값이 print되는가? 그 이유와 함께 설명하라. (5점)

```
int i;
i = 131,072 ; //  1024*128 = 131,072 (2^17)
System.out.println(i) ;
```

5. 아래와 같은 C 프로그램이 32-bit 컴퓨터상에서 수행되었을 때 (a) ~ (d) 문장에 의하여 프린트 되는 값을 써라 (단, word alignment를 한다고 가정한다. print되는 형식에 주의 할 것.) (각 4점 x 4 = 16점)

```
char aaa ;
union utag {
  char b[2]; char *c ; char a[8] ;
} ;
struct aa {
  float j; char aaa; union utag kk ;
} test[10][10], lala;
main() {
  printf("%x %x\n", &aaa, &(test[3][4]));  /* 프린트 결과는 4720,4730 */
  test[3][4].kk.c = &aaa ;
  printf("%d\n", sizeof(lala));                                  /* (a)  */
  printf("%x %x \n",&(test[3][4].kk),&(test[3][4].kk)+1);   /* (b) */
  printf("%x %x \n",(test[3][4].kk.c),(test[3][4].kk.c)+1); /* (c) */
  printf("%x %x \n",&(test[3][4].kk.c),&(test[3][4].kk.c)+1);/* (d) */
}
```

6. 다음과 같은 C 언어 프로그램과 Linux (Pentium PC용)상에서 컴파일된 어셈블리 코드를 보고 다음의 물음에 답하라. (각 4점 x 4 = 16점)

(a) C 언어에서 "=" 와 "+" operator의 associative rule이 어떠한지 어셈블리 코드를 이용하여 설명하라. (4점)

(b) C 언어에서 "+" operator의 operand evaluation order가 어떠한지 어셈블리 코드를 이용하여 설명하라. (4점)

(c) coercion이 어떤 방식으로 일어나는지 어셈블리 코드를 이용하여 설명하라. (4점)

(d) 이 프로그램의 프린트된 결과를 써라 (4점)

```
int a,b,c ; float k ;
int sub(int x) {
   b=b+2;
   return(0);}
main() {
  a=b=c=1; a = a + b + c;
  k = a + b + sub(c) ;
  printf("%f\n", k); }
```

```
.file   "test2.c"
.text
        .align 4
.globl sub
        .type    sub,@function
sub:
        pushl   %ebp
        movl    %esp, %ebp
        addl    $2, b
        movl    $0, %eax
        popl    %ebp
        ret
.Lfe1:
        .size    sub,.Lfe1-sub
                .section
.rodata
.LC0:
        .string "%f\n"
```

```
.text
        .align 4
.globl main
        .type    main,@function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        movl    $1, c
        movl    $1, b
        movl    $1, a
        movl    b, %eax
        addl    a, %eax
        addl    c, %eax
        movl    %eax, a
        subl    $12, %esp
        pushl   c
        call    sub
        addl    $16, %esp
        movl    %eax, %edx
        movl    b, %eax
        addl    a, %eax
        addl    %edx, %eax
        pushl   %eax
        fildl   (%esp)
        popl    %eax
        fstps   k
        subl    $4, %esp
        flds    k
        leal    -8(%esp), %esp
        fstpl   (%esp)
        pushl   $.LC0
        call    printf
        addl    $16, %esp
        leave
        ret
.Lfe2:
        .size    main,.Lfe2-main
        .comm   a,4,4
        .comm   b,4,4
        .comm   c,4,4
        .comm   k,4,4
        .ident  "GCC: (GNU)"
```

7. 오른쪽과 같은 C-like 언어에서 (a) static scoping rule을 사용하는 경우와 (b) dynamic scoping rule을 사용하는 경우에 각각 print되는 값은 무엇인가 ? (각 5점 x 2 = 10점)

```
int a, b ;
int p(void) {
  static int a = 0, p = 0 ;
  a = a+1; b = 1; p = p+2;
  return(p) ;
}
void print(void) {
  printf("a=%d,b=%d\n",a,b);
}
void q(void) {
  int b;
  a = 3 ; b = 4 ;
  print();
}
void main(void) {
  a = p()+p() ;
  q() ;
}
```

8. 어떤 프로그램을 수행시키기 위하여 메모리상에 적재하면, Code 영역, Data 영역, Stack 영역, Heap 영역 등이 메모리 상에 적재/생성된다. (13점)

(a) 위의 각 메모리 영역에 저장되는 내용은 무엇인가 ? (4점)

(b) Stack 영역은 LIFO (Last-in-First-Out)형식으로 관리 된다. 그 이유를 설명하라. (3점)

(c) Heap 영역에 할당된 변수들의 저장 공간은 언제 할당되고, 언제 회수 되는가 ? (6점)

9. (Python 프로그래밍) 리스트 A의 각 원소는 또 다시 리스트이다. A 내의 각 리스트 안에 있는 원소들을 제곱한 합으로 리스트 A를 구성하는 프로그램을 작성하시오. (7점)

```
A = [[1,2,3], [4,5], [1,3,5,7], [1,1,1,1,1]]
print('original A :', A)
/*
여기를 채워라
*/
print('changed A :', A)
```

수행 결과

```
original A :[[1,2,3], [4,5], [1,3,5,7], [1,1,1,1,1]]
changed A : [14, 41, 84, 5]
```