Assignment 2

Principles of Programming Languages

Number of questions: 10

Total Score: 100 points

Due day: 11:00 PM 25th May 2023

P.S.:

(1)You need to type your answers in an excel file and then submit your file to the TAs through new ee-class. The 作業 section of this course in the new ee-class has a template of this excel file.

(2)Your file name should have this format 學號_姓名_PPL_Assignment_2.xlsx.

(3) Without reasonable reasons, late submission will not be accepted.

(4) Copying other student's answers is strictly prohibited.

---

(1) (9 points)

What follows is a C program excerpt.

```
void bar()
{  int  tuv[10][10];
        :
        :
}
```

(a) If we do not consider whether we can change the value of variable `tuv`, what other properties does the value of `tuv` have?

(b) Could we change the value of `tuv`?

(c) Could we change the value of `tuv[1]`?

Ans.

(2) (12 points)

What follows is a Java program excerpt.

```
class Circle
{
  int setVariable(int s)
  {  int r;

     r=6;
     return s+r;
  }
}


public class xyz{
     :
  public static void main(String [] args)
  {  Circle  tuv[][]=new Circle[10][10];
        :
  }
}
```

(a) What data type does `tuv` have?

(b) What data type does `tuv[1]` have?

(c) Could we change the value of `tuv[1][1]`?

Ans.

(3) (12 points)

What follows is a Java program excerpt.

```
class Circle
{
  int setVariable(int s)
  {  int r;

     r=6;
     return s+r;
  }
}

public class Geometry{

     :

  public static void main(String [] args)

  {

    String t=new String("Welcome");      // location 1

    String s=new String("Welcome");      // location 2

    Circle  shape[][]=new Circle[3][3]; // location 3

         :

  }

}
```

(a)

Assume "Welcome" does not appear in any other statement before location 1, how many new String objects will be created by the Java statement at location 1?

(b)

Assume "Welcome" does not appear in any other statement before location 1, how many new String objects will be created by the Java statement at location 2?

(c)

How many objects will be created by the Java statement at location 3?

(d)

How many variables will be created by the Java statement at location 3?

Ans.

(4) (12 points)

C, C++, and Fortran do not perform range checking of array subscripts. Assume a C compiler allocates memories for arrays declared in the same declaration statement in adjacent areas and memory allocation stars from the leftmost array to the rightmost array. For example for the following declaration statement, `int t[10], u[10], v[10];`, the compiler allocates memory for array `t` first. The memory of array `u` is right after the memory of array `t`. The memory of array `v` is right after the memory of array `u`. Hence, the addresses of array `t` elements are larger than the addresses of array `u` elements. And the addresses of array `u` elements are larger than the addresses of array `v` elements. Assume the compiler utilizes row major order to store elements in an array. What follows is a C program.

```
#include<stdio.h>

main()

{ int a[10][10], b[10][10], c[10][10];

  int i,j;


  for(i=0;i<10;i++)

    for(j=0;j<10;j++)

    {

      a[i][j] = 1;

      b[i][j] = 2;

      c[i][j] = 3;

    }

    b[17][9]=4;     /*location 1*/

    b[17][16]=5;    /*location 2*/

    b[-3][7]=6;     /*location 3*/

    b[-3][-6]=7;    /*location 4*/

}
```

(a) The value of which array element will be changed by the statement at location 1?

(b) The value of which array element will be changed by the statement at location 2?

(c) The value of which array element will be changed by the statement at location 3?

(d) The value of which array element will be changed by the statement at location 4?

Ans.

(5) (8 points)

If dynamic scope is used in the following program, at location 10 (a) what is the value of variable `a`? (b) what is the value of variable `b`?

```
int car()
{ int e, f;

  e=7;
  f=8;
  return e+c+a;   //location 1
}
int bar()
{ int c=4, d=5, e=6;
  c=car()+b;  //location 2
  a=5;        //location 3
  e=9;        //location 4
  return c;   //location 5
}
int main()
{ int a, b, c;

  a=1;       //location 6
  b=2;       //location 7
  c=3;       //location 8
  b=bar()+c;  //location 9
  return 1;   //location 10
}
```

Ans.

(6) (8 points) In the following **C** program excerpt,

(a) what problem does this program have?

(b) why does this problem happen?

```
#define BufferSize 60
char *poi=”The summer break is around the corner.”;
char sentence[BufferSize];
int ppp(char *s, char *d, unsigned len)
{ unsigned i;
  for( i=0 ;i<len; ++i )
  {
    *(d+i)=*(s+i);
  }
}
void goo(char *s, char *d, int length)
{
   if(length<BufferSize)
     ppp(poi,sentence,length);
}
```

Ans.

(7) (8 points) What follows is a C program.

```
#include<stdio.h>
#define  SMALL_NUMBER_1  0.000007
#define  SMALL_NUMBER_2  0.000009
```

```
main()
{
  float  a, b, c, d, e;

  a=SMALL_NUMBER_1;
  b=c=SMALL_NUMBER_2;

  printf("a=%f\n",a);
  d=(a/(b*b*b*b*b*b*b*b*b*b));                /*location 1*/
  e=(a/(b*b*b*b))*(c/(b*b*b*b))*(c/(b*b));  /*location 2*/
  d=d*c*c;                                   /*location 3*/
  printf("(1)d = %f\n",d);                   /*location 4*/
  printf("(2)e =%f\n",e);                    /*location 5*/
}
```

(a) From the point of view of mathematics, the computation executed by the statement at location 1 and the statement at location 3 together is equivalent to the computation executed by the statement at location 2. In other words, the expression consisted of the statements at locations 1 and 3 is equivalent to the expression consisted of the statement at location 2. Hence, when the statement at location 3 is completed, variable `d` and variable `e` are supposed to have the same value. But statements at location 4 and location 5 show that variables `d` and `e` have different values? Explain the reason.

(b) Write a C program as the above one to find two equivalent expressions with different evaluation orders that have different values. (P.S.: You must paste your program and its execution result on your answer sheet.)

Ans.

(8) (12 points)

What follows is a C program named `test_address.c`.

```
#include <stdio.h>

int t;

int *p;

main()

{ int q ;


   t=987 ;

   p=&t;

   printf("Hello \n");  /* location 1 */

}
```

Assume after program `test_address.c` is compiled by a C compiler, the addresses of variables `t` and `p` are 678 and 123 respectively. Then after the program is executed and the execution flow arrives at **location 1**, what are the values of `&p`, `p`, and `*p` respectively?

Ans.

(9) (9 points) Consider the following program:

In the following **C** program excerpt,

(a) What is the result printed out by the statement commented as ``Location 6''?

(b) What is the result printed out by the statement commented as ``Location 11''?

(c) What mistake does it make?

```
void bar()

{ char *p,*q;                /*Location 1*/


   q=malloc(1);               /*Location 2*/

   p=q;                       /*Location 3*/

   *q='h';                    /*Location 4*/

   *p='e';                    /*Location 5*/

   printf("%c",*q);           /*Location 6*/
```

```
   q=malloc(1);              /*Location 7*/
   p=q;                      /*Location 8*/
   *q='r';                   /*Location 9*/
   *p='o';                   /*Location 10*/
   printf("%c",*q);          /*Location 11*/
}
```

Ans.

(10) (10 points) Consider the following program:

```
procedure Main is
   X, Y, Z : Integer;
   procedure Subl is
     A, Y, Z : Integer;
   begin -- of Subl
    . . .
   end;   -- of Subl
   procedure Sub2 is
     A, B, Z : Integer;
   begin -- of Sub2
    . . .
   end;   -- of Sub2
   procedure Sub3 is
     A, X, W : Integer;
   begin -- of Sub3
    . . .
   end;   -- of Sub3
begin  -- of Main
 . . .
end;   --  of Main
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

(a) `Main` calls `Subl; Subl` calls `Sub2; Sub2` calls `Sub3.`

(b) `Main` calls `Subl; Subl` calls `Sub3.`

(c) `Main` calls `Sub2; Sub2` calls `Sub3; Sub3` calls `Subl.`

(d) `Main` calls `Sub3; Sub3` calls `Subl.`

(e) `Main` calls `Subl; Subl` calls `Sub3; Sub3` calls `Sub2.`

Ans.