SECJ 1013 SEMESTER 1 2020 / 2021

PROGRAMMING TECHNIQUE 1

LAB EXERCISE 5 (4%)

NAME: MD MEHEDI HASAN

MATRIC NUMBER: A20MJ4005

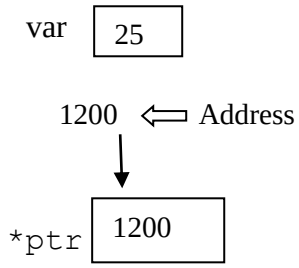
1) State whether the following declarations are valid or invalid. Give reasons for the invalid declarations and draw memory layout for the valid declarations.

i.
```
int var = 25;
int *ptr = &var;
```

Answer:

The declarations are valid

`int var = 25;` [suppose address of var is 1200]
int *ptr = &var; [address of var stored into pointer ptr]



ii.
```
int var = 30;
int* ptr = var;
```

Answer:

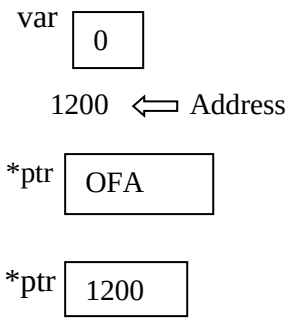
The declarations are invalid because integer value cannot assign directly to pointer

iii.
```
int var, *ptr;
ptr = &var;
```

Answer:

The declarations are valid

[suppose address of var is 1200]
[default value of var is 0]
[OFA- default address]



iv.
```
float fvar;
int *ptr = &fvar;
```

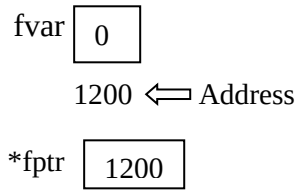
Answer:

The declarations are invalid because cannot assigned address of float variable to integer pointer

v.
```
float fvar, *fptr = &fvar;
```

Answer:

float fvar, *fptr= &fvar
[suppose address of fvar is 1200]



vi.
```
int *ptr = &var;
int var = 25;
```

Answer:

The declarations are invalid because Variable var address assigned to pointer before variable var declaration.

vii.
```
double* dptr1, dptr2;
double dvar = 25.2;
dptr1 = &dvar;
dptr2 = &dvar;
```

Answer:

The declarations are invalid because Address of dvar cannot assigned to non-pointer variable dptr2

Here only dptr1 is pointer variable and dptr2 is non-pointer variable

2) Determine the output for code segment below:

```
int numbers[5] = {2,4,6,8,10};
int *numPtr;

numPtr = numbers;
cout << *(numbers + 3) << endl;
cout << *(numbers) << endl;

for (int count=0; count < 2; count++)
{
    cout << *numPtr  << "\t";
    numPtr++;
}
cout << endl;

for (int count=0; count < 2; count++)
{
    numPtr--;
    cout << *numPtr << "\t";
}
```

Answer:

output:

```
8
2
2               4
4               2
```

3) Determine the output and draw a memory layout (or memory allocation) of the pointers and variables for code segment below:

Note: Draw a memory layout that represents C++ statement line by line

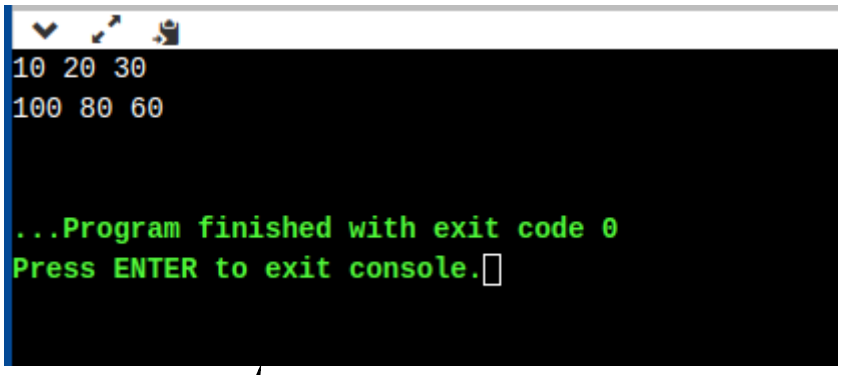
```
int  x = 10, y = 20, z = 30;
int *ptr;

cout << x << " "  << y << " " << z << endl;
ptr = &x;
*ptr *= 10;
ptr = &y;
*ptr *= 4;
ptr = &z;
*ptr *= 2;

cout << x << " "  << y << " " << z << endl;
```

Answer:

output:

10 20 30

100 80 60



x 10 x

ptr (ptr=Rx)
*ptr *=10
(x) * =10 =100

y 20

⇧

ptr (ptr=Ry)
*ptr *=4
(y) * =4 =80

y 80

z

ptr (ptr=Rz)
*ptr *=2
(z) * =2 =60

z | 60

1. initially value of x = 10
ptr = &x (ptr is now pointing to address of x)
(*ptr means ptr is pointing to value stored at x i.e 10 in this case)
(now *ptr and x becomes equivalent or same.modifying *ptr will modify value of x)
therefore,
*ptr *= 10 (will modify value of x to 100 and store it in x)

2. initial value of y = 20
ptr = &y (ptr is now pointing to address of y)
(*ptr means ptr is pointing to value stored at y i.e 20 in this case)
(now *ptr and y becomes equivalent or same.modifying *ptr will modify value of y)
therefore,
*ptr *= 4 (will modify value of y to 80 and store it in y)

3. initial value of z = 30
ptr = &z (ptr is now pointing to address of z)
(*ptr means ptr is pointing to value stored at z i.e 30 in this case)
(now *ptr and z becomes equivalent or same.modifying *ptr will modify value of z)
therefore,
*ptr *= 2 (will modify value of z to 60 and store it in z)

4) Write code segments to perform the following tasks.

i. Declare an integer array, numList with size of 8

Answer:

```
int numList[8];
```

ii. Declare an integer pointer, iPtr

Answer:

```
int * iPtr;
```

iii. Use for loop to read 8 integers from the keyboard

Answer:

```
for (int i=0; i<8; i++) {
    cin >> numList[i];
}
```

iv. Use array as pointer constant to display the numbers

Answer:

```
for (int i=0; i<8; i++) {
    cout << *(numList + i);
}
```

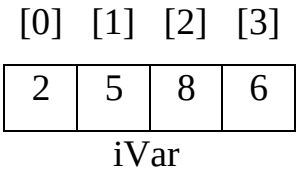
5) Write two statements to free dynamically allocated array and double which are declared as follows:

```
int *iPtr = new int [100];
double *dPtr = new double;
```

Answer:

```
delete [] iPtr;
delete dPtr
```

6. Starting address of the following array named iVar is 0xFEC07.



What is the output that will be displayed based on the following statements?

i. `cout << iVar;`

Answer:

cout << iVar;

The name of an array acts as pointer to the array. So, the name of the array points to the location in memory where the first element of the array is stored.

So, this statement will print the starting address of the array iVar

So 0xFEC07 will be printed

ii. `cout << iVar [0];`

Answer:

cout << iVar[0];

This statement prints the value stored at the 0th index/starting location of the array iVar

The element stored at the 0th index is 2.

So, 2 will be printed.

iii. `cout << *iVar;`

Answer:

cout << *iVar;

As I said earlier that the name of the array acts a pointer to the starting location of the array.

So iVar acts as the pointer the 0th element.

*iVar gives the value at the address pointed by iVar

The value stored at the starting location is 2

So, 2 will be printed.

iv. `cout << *(iVar + 2);`

Answer:

cout << *(iVar+2);

iVar contains the address of the 0th element

(iVar + 2) is the address of the element at index 2

*(iVar + 2) is value stored at index 2

The element stored at index 2 is 8

So, 8 will be printed