

You may be asked to demonstrate/explain your work to the tutor, if you are absent/unavailable or fail to demonstrate properly, zero marks will be awarded. Always

Lab exercises

1. Explain the following in your own words using examples and diagrams

- a. Variables
- b. Arrays
- c. Multi-dimensional arrays
- d. Pointers
- e. Double pointers

2. Write a program for the below set of statements and give memory snapshots after each of these sets of statements is executed:

- a) `int a=1, b=2, *ptr;`
...
`ptr = &b;`
- b) `int a=1, b=2, *ptr=&b;`
...
`a = *ptr;`
- c) `int a=1, b=2, c=5, *ptr=&c;`
...
`b = *ptr;`
`*ptr = a;`
- d) `int a=1, b=2, c=5, *ptr;`
...
`ptr = &c;`
`c = b;`
`a = *ptr;`
- e) `double x=15.6, y=10.2, *ptr_1=&y, *ptr_2=&x;`
...
`*ptr_1 = *ptr_2 + x;`
- f) `int w=10, x=2, *ptr_2=&x;`
...
`*ptr_2 -= w;`

Exercises:

3. Show the contents of the variables that have changed after executing each of the following codes (Draw table or figure to explain).

- a) `p1 = &i;`
`*p1 = 8;`
- b) `p1 = &m[0];`
`p1 = p1 + 2;`
`*p1 = *p1 + 8;`
- c) `p1 = &i;`
`p3 = m;`
`*(p3 + 1) = *p3 + *p1;`
- d) `p1 = m + 2;`
`p3 = p1 + 1;`
`i = *p1 + *p3;`
- e) `for(i = 0; i <= 4; ++i)`
`q[i] = m + i;`
`for(i = 0; i <= 4; ++i)`
`*(a + i) = *q[i] + a[i];`
- f) `for(i = 4; i >= 0; --i)`
`{`
`q[i] = &m[4 - i];`
`*p2 = *q[i] + *(a + i);`
`}`

4. Assume that an array `g` is defined with the following statement:

```
int g[]={50,10,6,81,130,4,9};
int *ptr1=&g[0], *ptr2=&g[3];
```

Give a **diagram** of the memory allocation, including the array values. Also indicate the offset values from the initial value in the array. Using this information, give the value of the following references:

1. `*g`
2. `*(g+1)`
3. `*g+1`
4. `*(g+5)`
5. `*ptr1`
6. `*ptr2`
7. `*(ptr1+1)`
8. `*(ptr2+2)`

5. Assume that an integer array **x** is defined by the following statements:

```
int x[2][4]={{1,8,7,6},{2,4,-1,0}}, *xptr=&x[0][0];
```

Draw a memory allocation **diagram**, and give the value indicated by each of the following references:

1. *xptr
2. *(xptr+2)
3. *xptr + 2
4. *(xptr+1) + *(xptr+3)

6. Write a simple function that returns a pointer.

It is advised to avoid spaces in the names of your files and folders.