Aum Amriteswaryai Namah

19CSE205 – Program Reasoning – Assignment – 2

1. Write a C program which should include the following statement, compile the program and write down the error you get. Ensure no other errors are in the program before trying this.
   - ◦ int 23ab;

2. Now change the statement as follows. Note there is a comma instead of semicolon at the end. Compile and write down the error you get.
   - ◦ int ab23,

3. Replace comma by semicolon.
   - ◦ int ab23;

     Add the following statements below the declaration, **one at a time**, compile-run the program and note down your observations.

   a. ab23 = 25;
   b. ab23 = 25.25;
   c. ab23 = 'Z';
   d. ab23 = "hello";
   e. int ab23; // i.e. another declaration
   f. int ab23[5]; // i.e. declare array variable with same name
   g. void ab23(); // i.e. declare function prototype with same name .

   What are your inferences from this experiment?

4. Declare an array as follows and assign a value.
   a. int arr[10]; arr[10] = 21;
      Does it produce compiler error or runtime error? What type of error it is?

5. Write a basic program with pointers as directed below.
   a. Declare a pointer to an integer variable ptr.
      int * ptr;
   b. Use malloc to dynamically allocate memory for ptr.
      ptr = (int *) malloc(sizeof(int));
   c. Assign an integer value to the memory pointed to by ptr.
      *ptr = 10;
   d. Print the value pointed to by ptr to the terminal.
      printf("%d\n", *ptr);
   e. Free the ptr memory.
      free(ptr);

6. **Perils of pointers**: Variations of Qn 5 to simulate the problems due to mishandling of pointers. All these are semantic errors.
   a. **A case of null pointer**: Access value without allocating the memory

   i. Perform steps a and c (i.e. without b).
   ii. Note down the error.

b. **Another case of null pointer**: Access value after freeing the memory
   i. Perform steps a, b, c, e and then d.
   ii. Note down the error.

c. **A case of memory leak**: Allocating memory without freeing. (Try this last after answering all questions, since system will gradually slow down and eventually crash.)
   i. In an infinite loop, do steps a, b, c and d (i.e. without e).
   ii. Run the program for as long as it can.
   iii. The program + system will crash after some time. Restart your computer.

d. **A case of not allowing memory leak**: Allocating memory with freeing
   i. In an infinite loop, do steps a, b, c, d and e.
   ii. The program should run forever without crashing.
   iii. Press CTRL+C to stop execution.

e. **A case of lost pointer and memory leak**: Re-assigning a pointer to another location

```
int * p = (int *) malloc( sizeof(int) ); // p points to a location_1 in memory
*p = 5;
int * q = (int *) malloc( sizeof(int) ); // p points to a location_2 in memory
*q = 10;
p = q; // p is reassigned and both p and q point to location_2
```

   i. The access to location_1 is lost. It is impossible to retrieve the value 5.
   ii. It can’t be freed either since the pointer is lost. This leads to memory leak.

f. **Another case of lost pointer**: Re-assigning a pointer to a new memory

```
int * p = (int *) malloc( sizeof(int) ); // p points to a location_1 in memory
*p = 5;
p = (int *) malloc( sizeof(int) ); // p reassigned to a new location (location_2)
*p = 10;
```

   i. The access to location_1 is lost. It is impossible to retrieve the value 5.
   ii. It can’t be freed either since the pointer is lost. This leads to memory leak.

g. **A case of dangling pointer**: Freeing up a pointer when another is accessing the same location.

```
int * p = (int *) malloc( sizeof(int) ); // p points to a location_1 in memory
int * q = p; // Both p and q not point to location_1
free(p); // p frees location_1. The runtime system claims it.
*q = 10; // Can’t access location_1 since it is no more program memory
```

   i. In short, q points to a location which does not legally belong to the program
   ii. Note down the error

h. **A case of messing up with pointer**: incorrect type casting leads

```
long * ptr = (long *) malloc( sizeof(int) ); // 4 bytes allocated
*ptr = 10; // This will access 8 bytes of memory
```

   i. Out of 8 bytes, only first 4 can be legally accessed.
   ii. The runtime system will report an error when trying to assign value 10.
   iii. Note down the error

7. In the light of what you have learnt about lexical, syntax and semantic errors, apply them to English language. Determine the type of the error in the following
    a. This statement has tow errorrs.
    b. I is going for lunch.
    c. Nita told Gita that it is her bag.
    d. This is Ashish. She is the topper.

8. Answer the following questions.
    a. What is the difference between lexical and syntax error?
    b. What is the difference between syntax and semantic error?
    c. What is the difference between compile time detectable semantic error and runtime detectable semantic error?
    d. How will you differentiate logical error and semantic error?

9. Write a program for computing absolute value of an integer. abs(x) = x if x is positive, -x, if x is negative. Don't use the library call. How many test cases are required to check each execution path the program can take?

10. Write a program that takes x and y as input, and prints which quadrant (x,y) belongs to. For instance, given (4,-3) as input, it should print Q4. Provide exhaustive set of test cases that will ensure every case is covered.