Review Questions – Set #1

Question 1.

Provide a brief answer for each of the following.

a) Identify two main tasks performed by a C compiler.
b) Explain one advantage of using pointers.
c) Explain one disadvantage of using global variables.

Question 2.

Each of the following C programs is associated with one of three scenarios: a) failure to compile due to a syntax error, b) segmentation fault at runtime, and c) normal run. Choose the correct answer and, in the case of "Normal run", provide the output that would be printed on the screen. You can assume that all the required libraries are included.

Program #1

```
int main(){
  int x;
  scanf("%d", &x);
  square = x * x;
  printf("%d\n", square);
  return 0;
}
```

a. Syntax error b. Segmentation fault c. Normal run	Output:

Program #2

```
int main(){
  double x=7.0, y=2.0;
  int div = x/y;
  printf("%d\n", div);
  return 0;
}
```

a. Syntax error b. Segmentation fault c. Normal run	Output:

Program #3

```
int main(){
  char c = 'A'; //ASCII 65
  char d = c + 1;
  printf("%c\n", d);
  return 0;
}
```

a. Syntax error b. Segmentation fault c. Normal run	Output:

Program #4

```
int main(){
  int a[3]={10,20,30};
  int *p = &a[2]-2;
  printf("%d\n", *p);
  return 0;
}
```

a. Syntax error b. Segmentation fault c. Normal run	Output:

Program #5

```
void foo(int x){
  if(x/2 == 1) x = 0;
  else x=1;
}

int main(){
  int x=3;
  foo(x);
  printf("%d\n", x);
  return 0;
}
```

a. Syntax error
b. Segmentation fault
c. Normal run

Output:

Program #6

```
void foo(int *p){
  p = NULL;
}

int main(){
  int x=2;
  int *p = &x;
  foo(p);
  printf("%d\n", *p);
  return 0;
}
```

a. Syntax error
b. Segmentation fault
c. Normal run

Output:

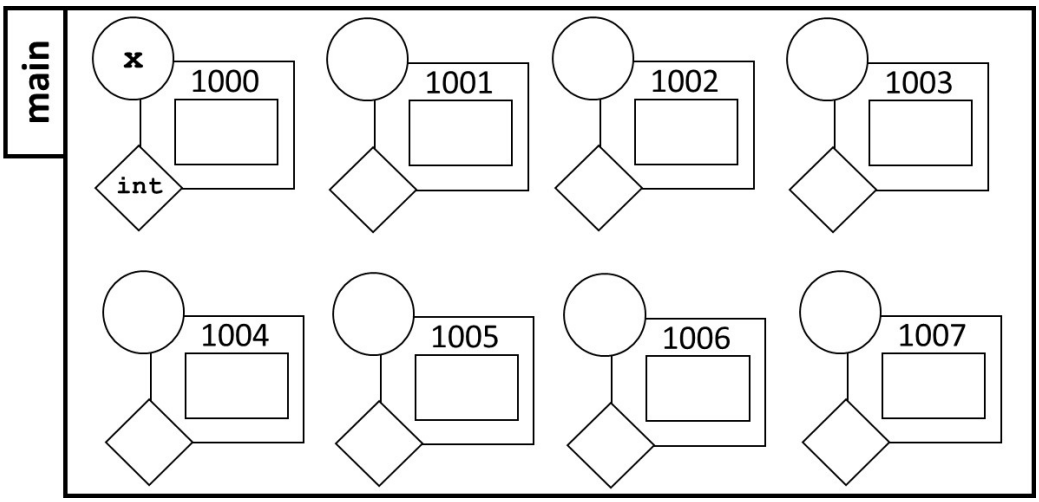
Question 3.

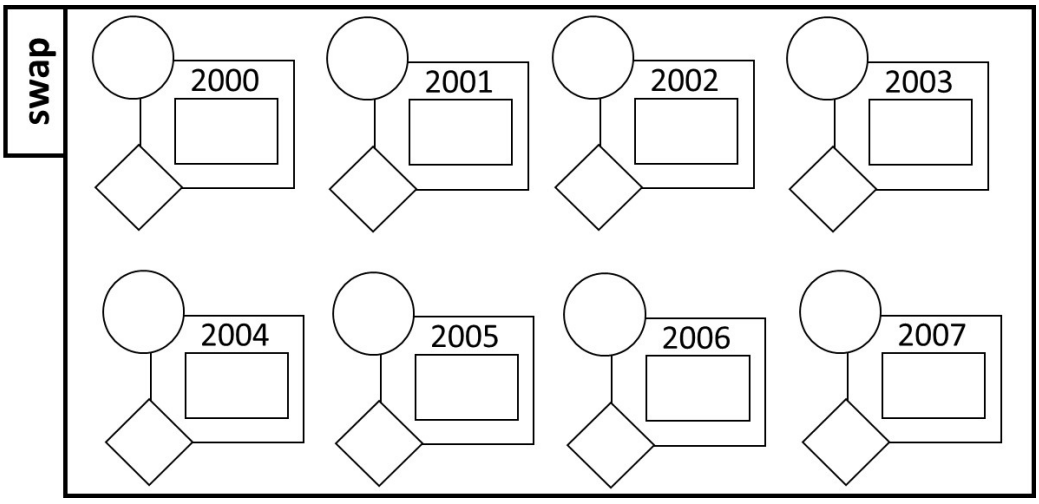
Consider the following code:

```
int main(){
  int x = 4;
  int a[3] = {5, 6, 7};
  int *p = a+1;
  *p = 8;
  swap(&x, p);
  return 0;
}
```

```
void swap(int *p, int *q){
 int temp = *p;
 *p = *q;
 *q = temp;
 printf("Done swapping\n");
}
```

You are required to show the memory contents when the `printf` statement in function `swap` is reached during execution. Each box in the layout provided below consists of 4 components: an address (e.g. 1000, 1001, etc), a label (circle), a data type (rhombus), and a value (rectangle). You need to specify the label, data type, and final value for each box you use (the label and the data type of the first box are provided). Also, note that variables should be assigned to boxes based on the order they are declared and each variable should be assigned to the next available box. You might not need all the provided boxes.





Question 4.

The following code is supposed to read two arrays of integer values from the user and determine the maximum number that appears in both arrays. If there is no common element between the arrays, a corresponding message will be printed. To achieve this, you are required to implement functions `search` and `maxCommon`, and fill in the required sections within the `main` function.

```
//This function returns 1 if A contains val, and 0 otherwise.
//len represents the length of A
int search(int val, int A[], int len){
     //implement this function




}

//If A and B have no common elements, this function returns 0.
//Otherwise, it returns 1 and stores the maximum common element using
//max. lenA and lenB represent the lengths of A and B respectively.
int maxCommon(int A[], int lenA, int B[], int lenB, int *max){
     //implement this function




}

//The number of elements to store in each array is provided by the
//user. You can assume that it will not exceed 100.
//Fill in the blanks.
int main(){

     __________ A1[100], A2[100], len1, len2;

     printf("Enter the length of the first array: ");

    scanf("%d", &len1);

    printf("Enter the elements of the first array\n");

    for(int i=0;____________;i++)

          scanf("%d", ___________);


     printf("Enter the length of the second array: ");
```

```
    scanf("%d", &len2);
    printf("Enter the elements of the second array\n");
    for(int i=0;____________;i++)
        scanf("%d", __________);
    int max;
    int result = maxCommon(_______________________________);
    if(_____________)
        printf("The arrays have no elements in common\n");
    else
        printf("The maximum common element is %d\n",____);
    return 0;
}
```