SECTION 1 - Multiple Choice Questions (1 mark each) – Total 5 marks:

Notes: To answer the questions in this section you can make the following assumptions:

- All necessary header files are included
- Size of `int` data type is 4 bytes, and size of `double` data type and pointers are 8 bytes.

1. What is the output of the following code fragment?

```
char* s1 = "apple";
char s2[]= "Victor";
int* p = malloc(5 * sizeof(int));
printf("%d %d %d %d", (int) sizeof(s1), (int) sizeof(s2),
                      (int) strlen(s2), (int) sizeof(p));
```

a. 5 6 5 8
b. 8 7 6 8
c. 8 8 4 8
d. 6 6 5 8
e. 6 6 6 4
f. None of the above.

2. What is the output of the following code fragment?

```
int a[] = {44, 88, 66, 22, 10, 12, 5};
int* p1 = a;
int* p2 = &a[6];
printf("%d", (int) (p1 - (p2 + 1));
```

a. 6
b. 5
c. -7
d. 7
e. None of the above.

3. What if anything is wrong with the following code fragment?

```
char code[6] = "Apple";
const char* csp = code + 1;
csp++;
*code = csp[0];
*csp = *code;
```

a. There is a compilation error on the second-last line of the code fragment.
b. There is a compilation error on the last line of the code fragment.
c. The fragment will cause the containing program to crash when it runs.
d. All of the above are correct
e. None of the above is correct.

4. Consider the following code fragment and select one of the following answers. Assume that the user enters the following text on the keyboard, then presses the Enter key: **`52.75  12.625x`**

```
double d = 0.125;
int i = 42, ns = 99;
printf("please enter two numbers: ");
ns = scanf("%d %lf", &i, &d);
printf("%d %d %f\n", ns, i, d);
```

a. The program output will be: `0 42 0.125000`
b. The program output will be: `1 52 0.125000`
c. The program output will be: 2 52 0.750000
d. The program output will be: `2 52 12.625000`
e. None of the above.

5. What is the output of the following code fragment?

```
char course[] = "ENCM339F2016";
char* sp = course + 2 ;
while(*sp != '1'){
    printf("%c", *sp);
    sp = sp + 2 ;
}
```

a. ENM339F20
b. CM33F20
c. C3921
d. EC3921
e. None of the above.

SECTION 2 – Short Answer Questions – Total 12 marks:

1. **(4 marks)** The following partial C program should open a text file called `input.txt`, with several `int` values separated by whitespace, reads one `int` value at a time, and copy each value into a binary file, until the end of the input file is reached. You may assume that the data in the text file are all valid integers and that both files will open successfully without errors. Your job is to fill the blank spaces with correct C expressions.

```
 #include <stdio.h>
int main(void) {
   int number, nscan;
   FILE *fp1 = fopen("input.txt", "r");
   FILE *fp2 = fopen("output.dat", "wb");
   while (1) {
     nscan = fscanf(………fp1………… , "%d", ………&number………… );

     if (nscan != …1… ) break;

     fwrite ((void *)…&number……, sizeof(……int……), 1, ……fp2……);
   }

   fclose(........fp1.......... );

   fclose(........fp2.......... );

   return 0;
 }
```

2. **(2 marks)** Assume that the macros `FOO`, `BAR` and `QUUX` are not defined in `<stdio.h>`. What is the output of the following program?

```
#include <stdio.h>
#define FOO 47
#define QUUX
int main(void) {
#if defined(QUUX)
    printf("99 QUUX\n");
#ifndef FOO
    printf("FOO\n");
#endif
#if 1
    printf("BAR\n");
#endif
#endif
    return 0;
}
```

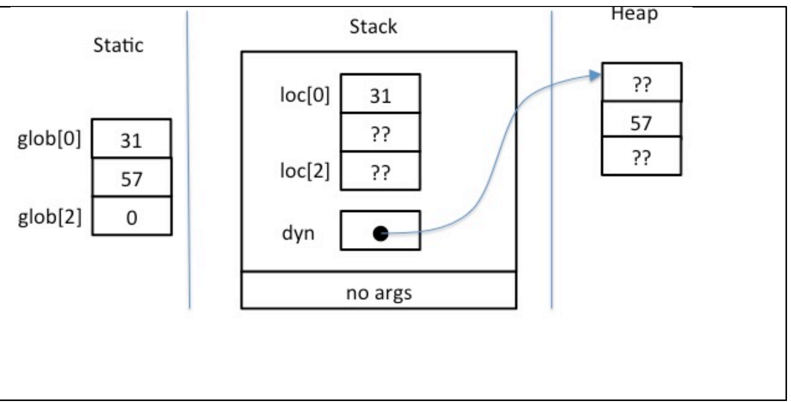
Write your answer in the following space:

```
99 QUUX
BAR
```

3. **(4 marks)** Assume that the call to `malloc` succeeds. Draw a memory diagram for `point one` in the space beside the code listing.

```
#include <stdio.h>
#include <stdlib.h>
int glob[3] = { 31, 57 };
int main(void) {
  int loc[3];
  int *dyn =
    malloc(3 * sizeof(int));
  loc[0] = glob[0];
  dyn[1] = glob[1];

  // point one
  return 0;
}
```



(2 marks) Write a `#define` macro with three arguments, called `IN_RANGE`, that produces a value of 1 if the value of the first argument is greater than or equal to the second argument and less than or equal to the third argument, and otherwise produces a value of 0.
Write your macro in the following space:

Three possible solutions:

```
#define IN_RANGE(x, y, z) (x) >= (y) && (x) <= (z) ? 1 : 0
#define IN_RANGE(x, y, z) (x) >= (y) && (x) <= (z)
#define IN_RANGE(x, y, z) (x) >= (y) ? (x) <= (z) ? 1 : 0 : 0
```

SECTION 3 – Functions – 26 marks:

Part a (8 marks): Write a function definition based on the given function interface comment.

```
int remove_outliers(int* x, int n, int min, int max);
// REQUIRES: n > 0. Elements x[0] ... x[n-1] exist.
// PROMISES: Elements less than min or greater than max are removed from the array.
//  Let rv be the return value. Then rv == number of elements that are >= min
//  and <= max, and x[0] ... x[rv-1] contain those elements.
// EXAMPLE: n is 5, original values of x[0] to x[4] are 5, -1, 4, 11, 10, min is 0,
//   max is 10. Then rv will be 3, x[0] to x[2] get 5, 4, 10, and it does not matter
//   what is left in x[3] and x[4].
```

Here are two possible solutions:

```
int remove_outliers(int* a, int n, int min, int max) {
    int j = 0;
    int i = 0;
    while (i < n){
        if(a[i] >= min && a[i] <= max) {
            a[j] = a[i];
            j++;
        }
        i++;
    }
    return j;
}
```

Common Mistakes for solution one:
Instead of using two indices, for example i and j to replace the removed outliers some students used a single index such as: a[i] = a[i+1]. This is a serious logical error if student are using a single loop approach to remove outliers

```
int remove_outliers1(int* a, int n, int min, int max) {
    int i, j, k=0;
    for(i = 0; i < n; i++){
        if(a[i] < min || a[i] > max){
            for (j = i; j < n-1; j++)
                a[j] = a[j+1];
            i--;
            k++;
        }
    }
    return i - k;
}
```

Common Mistakes:
When an outlier was removed by using a nested loop, the value of i was not updated properly (the statement, i-- within the if block in some codes was missing). Also some students didn't use proper logic to access the outlier.

Part b (6 marks): In mathematics, a Fibonacci integer sequence may have any two integers for its first two values. After that, each value is the sum of the previous two values. The Fibonacci numbers ...

0, 1, 1, 2, 3, 5, 8, 13, 21, …

are an example of such a sequence. Write a function definition for `is_fibonacci`.

```
int is_fibonacci(const int* x, int n);
// REQUIRES: n >= 3. Elements x[0] ... [n-1] exist.
// PROMISES: Return value is 1 if x[0] ... x[n-1] are the first n numbers of a
// Fibonacci integer sequence.  Otherwise the return value is 0.
```

```
int is_fibonacci(const int *x, int n) {
    for(int i = 2; i < n; i++){
        if(x [i] != (x[i-1] + x[i-2]))
            return 0;
    }
    return 1;
}
```

Common Mistakes:
One common mistake was, improper order of the function's return value. Another common misstate and more serious error was using wrong logics such as:

```
for(int i = 2; i < n; i++)
        if(x [i] != (x[i-1] + x[i-2]))
            return 0;
        else
            return 1;
```