Solutions for Midterm Exam
Fall 2018

Instructor: M. Moussavi

Note: This document contains solutions to all questions except for sections 4 and 6 (the AR diagram).

Please Notice: To answer the questions in this exam, you can always make the following assumptions:

- All necessary header files are included.
- Size of integer data type is 4 bytes, and size of double data type and pointers are 8 bytes.
- If you need, here are some ASCII values: `'A' == 65, 'a' == 97,and '0'(zero) == 48.`

SECTION 1 - Multiple Choice Questions (1 mark each) – Total 6 marks

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

```
const char* p = "Apple";
char s[]= "Act";
const char ch = *p;
printf("%lu %lu %lu %lu", sizeof(ch), sizeof(p), sizeof(s),strlen(p));
```

a. 1 1 3 6
b. 8 8 4 5
c. 1 8 4 5
d. 8 5 8 6
e. 1 8 4 6
f. None of the above.

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

```
 int a[] = {44, 88, 66, 22, 10};
 int* p1 = a;
 int* p2 = &a[5];
 long int L = p2-p1;
 printf("%ld", L);
```

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

3. Consider the following code fragment?

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

Select the best answer:

a. There is a compilation error on the last line of the code fragment.
b. There is a runtime error on second last line of the code fragment
c. Both of the above statements (a) and (b) are correct answers
d. None of the above answers are correct

4. In C++ member functions that retrieve the data (normally called getters), are declared as following examples:
`int getx() const;`
Which of the following statement are correct for `const` keyword in this position. Select the best answer:

a. It makes the member function as read-only function.
b. It will not allow the member function to change the data members of the class.
c. It is used only for the getter member function, and cannot be used by any other member functions.
d. a and b are both correct answers.
e. None of the above is a correct answer.

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

```
char course[] = "ENXM379F5016";
char* sp = course + 2 ;
while(*sp != '1'){
    printf("%c", *sp);
    sp = sp + 2 ;
}
```

a. NM379F50
b. XM379F50
c. X395
d. EC3921
e. None of the above.

6. Which one of the following statements is true about constructors in a C++ class:

a. Must have no return type.
b. Must have the same name as its class name.
c. Can be overloaded.
d. All of the above are true.
e. None of the above.

SECTION 2 – Short Answer Questions – Total 12 marks: By David

Question 1 (4 marks) – In this question you are supposed to fill in the blanks (dotted line) in the partial definition of the function `reverse`. This function is supposed to reverse the string `dest` in place and should return it. For your information, “in place” means you are supposed to shuffle characters in the `dest` itself, and not to copy them into another array. **Note:** You are **NOT** allowed to add any variables or change the existing code.

```
char* reverse (char* dest){
    char *p = dest + strlen(dest) -1;
    char *start = dest;

    while ( dest < p ) {

        char temp = *dest;

        *dest = *p;

        *p = temp;

         dest++;

          p--;
    }

    return start ;
}
```

Question 2 (8 marks) - Consider the following C `struct`:

```
typedef struct City{
    char cityName[20]; // name of a city
}City;

typedef struct Data{
    City city;          // city that data belongs to
    double rain[12];    // monthly rainfall of the city
} Data;
```

The following partial definition of function `read_data` is supposed to prompt the user to enter the name of a city and the rainfall data for 12 month, to be saved into struct Data that pointer x points to. Then, it is supposed to display the user’s entries on the screen. You should complete the function by filling the blanks.

```
void read_data (Data *x){
    int nscan, i = 0;
    printf("Please enter the name of the city: ");

    nscan = scanf("%s", x -> city.cityName);
    i = 0;
    printf("Please enter 12 monthly values for rainfall:\n");

    // reading 12 input data from user
    do {
        nscan = scanf( “%lf”, &x -> rain[i]);

       if(nscan != 1){
            printf("Invalid data.\n");
            exit(1);
        }

        i++;

    }while ( i < 12);

    printf("The rainfall data that you entered are:\n");

    i = 0;

    while (i < 12 ) {

        printf("%f\n", x -> rain[i]);

        i++;
    }
} // end of function
```

SECTION 3 – Functions – 23 marks

Part a (6 marks) Write a function definition to match the following function comment interface. This function must return one if **s** points to a valid string that represents a valid double number in C. Here is an example of a c-string that contains a valid double number:

‘6’	‘7’	‘9’	‘.’	‘5’	‘3’	‘\0’

```
int is_valid_real_number(const char* s);
// REQUIRES: s points to the beginning of a valid C-string.
// PROMISES: If s points to a C-string that represents a valid double number in C it returns one
// (a string that contains digits and only one decimal point).
// Otherwise if s points to an invalid double number or empty string returns zero.
// Note: The following seven numbers are examples of valid double numbers in c:
// 12234.999  .9999  8888.  77988  0  0000  00.000
// And the following four numbers are some examples of invalid double numbers:
//  12234..999   ..9999   8888..   23.9.   a77988
```

```
int is_valid_real_number(const char* s)
{
  int i=0;
  int count = 0;
  int sign = 0;
  if(s[0] == '\0') return 0;
  if((s[0] == '.' || s[0] == ‘-‘ || s[0] == ‘+’) && s[1] == '\0') return 0;
  if(s[0] == '-' || s[0] == '+') i++;

  while(s[i] != '\0')
  {
    if(s[i] == '.') count++;
    if((s[i] < '0' && s[i] > '9' && s[i] != '.' && sign == 0) || count > 1) return 0;
    i++;
  }
    return 1;
}
```

Part b (7 marks) – Write a C function that matches the following function prototype and interface comments.

```
double* reduce_size (double* x, int N, int M);
/* REQUIRES: M > 0, N > 0 and x points to a dynamically allocated array with N elements on the
 *          heap.
 * PROMISES: if M >= N does nothing. Otherwise reduces the number of elements of x from N to M,
 * preserves the values of the first M elements of x, and returns x.
 */
```

```
double* reduce_size (double* x, int n, int m){
     if(m >= n)
          return x;

     double *p = malloc(m * sizeof(double));

     if (p == NULL){
          printf("memory not available.");
          exit(1);
     }

     for(int i = 0; i < m; i++){
          p[i] = x[i];
     }

     free(x);
     x = p;
     return x;
}
```

Part c (10 marks): In the following space complete the definition of function `readBinary_writeText`, that opens a **binary input file** named by the C-string `inputFileName`, which contains several integer numbers (positive and negatives) and writes only the positive numbers from input file into a **text file** named by C-string `outputFileName`.

Note: you are not allowed to add more local variables or arguments to this function.

```
void readBinary_writeText(const char* inputFile, const char* outputFile)
{
    FILE *fp1;
    FILE *fp2;
    int number;

    if((fp1 = fopen (inputFile, "rb")) == NULL){
        fprintf(stdin, "Sorry cannot open the binary file %s.", inputFile);
        exit(1);
    }

    if((fp2 = fopen (outputFile, "wt")) == NULL){
        fprintf(stdin, "Sorry cannot open the binary file %s.", outputFile);
        exit(1);
    }

    if(fread( &number, sizeof(int), 1, fp1) !=1 && !feof(fp1)){
        fprintf(stderr, "reading from file %s failed", inputFile);
        exit(1);
    }

    while(!feof(fp1)) {
        if(number >= 0)
            fprintf(fp2, "%d\n", number);
        if(fread( &number, sizeof(int), 1, fp1) != 1 && !feof(fp1)){
            fprintf(stderr, "reading from file %s failed", inputFile);
            exit(1);
        }
    }

    fclose(fp1);
    fclose(fp2);


} // END OF FUNCTION
```

SECTION 4 – C structure – 5 marks:

In the following space draw an AR diagram for point one:

```
struct point {
 double x, y;
};

struct Rectangle {
 struct point* corner;
 double width, height;
};
```

```
void funct(struct Rectangle rec, struct Rectangle* p) {
  p ->width = rec.width;
  // point one
}

int main(){
  struct point* p = malloc(sizeof(struct point));
  struct Rectangle a = {NULL, 15, 66}, b = {p, 10, 20};
  funct(a, &b);
  return 0;
}
```

SECTION 5 – C++ classes – Total 4 marks   By Mahmood

Assume the following definition of class Point is in a header file called point.h.

```
class Point {
 private:
   double x, y;
 public:
   double getx()const;
   void setx(int v);
}; // END OF CLASS Point
```

Question a (2 marks): In the following space write the definition of member function `getx`.

```
double Point::getx() const
{

   return x;
}
```

Question b (2 mark
s): in the following space write the definition of function `setx`.

```
void Point::getx(int v)
{

   x = v;
}
```

SECTION 6 – Arrays and Pointers (10 marks): By David

Consider the definition of the following program and draw memory diagrams for points **one** and **two** in the given boxes to the right of the code:

```
#include <stdio.h>

char foo(const char *str, char arr[]);

void bar(int x[], int* y);


int main(void)
{
  char str[] = "ROSE";

  const char* exam = "Spring 18";

  int array[]={-4, 17, -300, 400};


  str[0] = foo(&exam[2], str + 2);

  bar(array, array + 2);


  return 0;
}


char foo(const char *str, char arr[])
{

  arr[0] = *(str-2);

  // point one

  return arr[0];
}

void bar(int x[], int* y)
{
  int z = *x;

  *x = y[0];

  y[0] = z;

  // point 2
}
```

Point 1

Point two