ECE 15, Winter 2022

Final

Sequence number	

Grade
/ 70

Last name	
First + middle name(s)	
PID	

Instructions:

- Do not look at the questions or start writing until it is announced you can do so.
- You cannot use any electronic devices.
- This exam is open book, open notes (provided they are on paper).
- Write only your actual answers on these pages. You may use the backside of the pages as well, but in that case clearly indicate where we can find your work.
- Use your own paper for scratch work (and you do not need to turn that in).
- Sign and date the academic integrity statement below.

Academic Integrity Statement:

Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and in so doing protect the validity of university intellectual work. For students, this means that all academic work will be done by the individual to whom it is assigned, without unauthorized aid of any kind.

By signing below, I certify that I have completed this exam in accordance with the academic integrity statement.

Signature: ____

Date: ____________________

▲

Question 1 (27 pts)

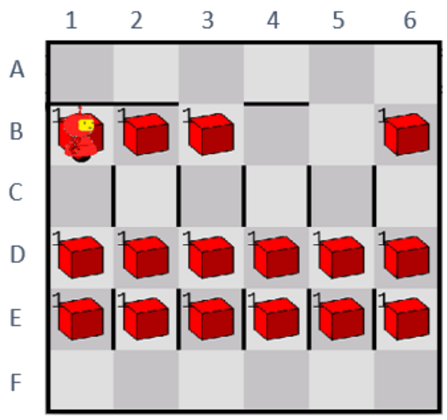
Complete the questions listed in the box to the right of the program. Important information:

- The programs should not contain any errors (run-time or compilation) unless it is specifically mentioned to check for possible errors.
- Assume this is a **64-bit system**, with **integers 4 bytes** and **doubles 8 bytes**.
- Use **?** if you want to denote a **garbage value**.

Part a

This is the start map with Karel's starting position (at coordinate B1, facing east). He has 2 items in his bag.

You may assume this program compiles and runs without issues (i.e., the settings file `set01.json` is correct and in the correct directory, all the map files are present, etc.).



```
#include <karel.h>

int main() {
  karel_setup("settings/set01.json");

  while (item_present()) {
     while (wall_to_left())
        move();
     move();
  }


  turn_right();
  move();
  while (wall_to_left() || item_present())
     move();


  if (facing_south() || !bag_empty() && !facing_south())
     turn_left();
  else
     turn_right();

  turn_off();
}
```

(a) When the program ends, what are Karel's coordinates?

(b) When the program ends, what direction is Karel facing (east, north, south or west)?

Part b

```c
#include <stdio.h>

int* process(int *x) {
   *x = *x + 1;
   x = x + 2;
   return x;
}

int main() {
   int *p;

   int a = 5;
   p = process(&a);
   printf("%d \n", a );
   printf("%d \n", *p );

   int b[4] = {7, 5, 3};
   p = process(b+1);
   printf("%d \n", p[0] );
   printf("%d \n", b[1] );
}
```

What gets printed?

Part c

```c
#include <stdio.h>

void process(char z[]) {
   *(z+1) = '\0';
   printf("%d \n", sizeof(z) );
}

int main() {
   char x[] = {'a','b','c'};
   char y[] = "def";

   printf("%d \n", sizeof(x)+1 );
   printf("%d \n", sizeof(y)+1 );
   printf("%d \n", sizeof(x+1) );

   process(x+1);
   printf("%s \n", x);
}
```

What gets printed?

Part d

```c
#include <stdio.h>

int main() {
   char x = 1.5;
   double y = 'd', z = 1.5;
   int k = 2;

   x += (int)2.75;
   printf("%1.2f \n", x + 0.5 );

   printf("%c \n", (char)(y + 3*z) );
   printf("%1.2f\n", (double)((int)3*z) );

   printf("%d \n", k > -2 );
   printf("%d \n", '\0'?--k:++k );
}
```

What gets printed?

Part e

```c
#include <stdio.h>

int a;

int process(int a) {
   a = a + a;
   return a;
}

int calc() {
   static int x = 3;
   x = a + x;
   return x;
}

int main() {
   printf("%d \n", a);
   a = 0;
   printf("%d \n", process(++a) );
   a = 1;

   if (1) {
      int a = 5;
      printf("%d \n", a + calc() );
      printf("%d \n", process(++a) );
      a = 2;
   }
   printf("%d \n", a + calc() );
}
```

What gets printed?

Part f

```
#include <stdio.h>

int main() {
   int i, j, a;

   for (i = 0; i < 2; i++) {
      j = 0;
      a = 1;
      while (j < 3) {
         j += 1;
         if (j == i) continue;
         if (j == 2) break;
         a *= 2;
      }
      printf("%d:%d \t", i, a);
   }
}
```

What gets printed?

Important Notes for Subsequent Questions

These notes apply to all questions that follow.

1. You are **not** allowed to use **global variables** or functions from a **string library**.
2. If you need **arrays** of variable length, you need to use dynamic memory allocation.
3. You are allowed and encouraged to use **helper functions**.
4. You will lose points if your code is **overly complicated** or showing a gap in understanding of the core material.

Question 2 (8 pts)

You must implement the functions **swap1**() and **swap2**() and call them in the code below.

swap1():	It must have a void return type. Referring to the code below, this function should swap the content of b and that of the first negative value of a.
swap2():	The functionality of swap2() is identical to that of swap1(), i.e., it should again swap the content of b and that of the first negative value of a (and the way you do that can be the same). This swap2() function should not have a void return type (which means that it should return something and assign that returned value to something).

In essence, swap1() and swap2() are simply two different versions of the same swap functionality that differ in how data is passed between the function and main().

The specific values for a and b in the sample code are only given as illustrations. The black box shows you the correct outputs for this example. Your code should be generic such that it works if we give a and b different values. You may assume that a will always have at least one negative value (for both swap1() and swap2()).

```c
#include <stdio.h>
#define DIM 6

int main() {
   int a[DIM] = {2,6,3,-1,2,9};
   int i, b = -5;

   // Call your swap1() function (single line of code)



   printf("%d \t", b);
   for (i = 0; i < DIM; i++)
       printf("%d ", a[i]);

   // Call your swap2() function (single line of code)



   printf("\n%d  \t", b);
   for (i = 0; i < DIM; i++)
       printf("%d ", a[i]);
}
```

```
 -1      2 6 3 -5 2 9
 -5      2 6 3 -1 2 9
```

Your task:

1. Implement the function **`swap1`**() below.
2. Implement the function **`swap2`**() below.
3. Complete the two missing lines in the code on the previous page. You cannot make any other modifications to `main()`.

Question 3 (7 pts)

You will be asked to implement the five functions, print_name(), set_pid(), init_grades(), set_grade() and get_street_letter(), that are called in the code below.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
   int number;                // The house number
   char street[50];           // The street name
} Address;

typedef struct {
   int pid;                   // The PID of the student
   char name[50];             // The name of the student
   int *grades;               // Dynamic array with student's grades
   Address address;           // The address of the student
} Student;

int main() {
   Student student1 = {0, "Jane Doe", NULL, {9500,"Gilman"}};

   // The function print_name() prints the name of the student
   print_name(student1);

   // The function set_pid() sets the pid of the student to
   // the value that is passed as the second argument
   int pid = 1234567;
   set_pid(&student1, pid);

   // The function init_grades() creates a new dynamic array that
   // is linked to by the grades field in the student struct. The
   // size of the dynamic array is given by the second argument.
   // All elements of the dynamic array should be initialized to 0.
   int num_grades = 4;
   init_grades(&student1, num_grades);

   // The function set_grade() sets the grade at a specific index
   // in the dynamic array to the value given as the third argument.
   int index = 1, value = 12;
   set_grade(student1, index, value);

   // The function get_street_letter() returns the first letter of
   // the street that is listed in the address of the student.
   char c = get_street_letter(student1);
}
```

Your task: Write the functions **`print_name`**(), **`set_pid`**(), **`init_grades`**(), **`set_grade`**() and **`get_street_letter`**(), such that when called as in the code on the previous page, they result in the behavior described by the comments.

Question 4 (13 pts)

You must implement three functions: **`next`**(), **`modify`**() and **`cut`**(). The functions are called as illustrated in the code below. The black box shows you the correct outputs for the test data.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
   char sentence[] = "go to the beach";
   char word[] = "overlord";
   char *p;
   int reset = 1;

   do {
        p = next(sentence, reset);
        reset = 0;
        if (p != NULL)
            printf("Next: %s \n", p);
   } while (p != NULL);

   modify(sentence);
   printf("Modify: %s \n", sentence);

   cut(sentence, word);
   printf("Cut: %s \n", sentence);
}
```

```
Next: go to the beach
Next: to the beach
Next: the beach
Next: beach
Modify: 7o to the beach
Cut: 7o to th
```

The three functions should implement the following functionalities:

`next`(): It takes in a string and each time it is called, it returns a pointer to the next word in the string. Words consist only of lower-case letters and are separated by single spaces. You may assume that the string does not change between consecutive calls of this function and your function should not modify the string. If the second argument (i.e., `reset`) is equal to 1, you start from the beginning of the string again. The function returns the NULL pointer when we have reached the end of the string (i.e., there are no more words left in the string).

`modify`(): It takes in a string and replaces the first letter by the last digit of its place in the alphabet (place in the alphabet starts at 1 for the letter **a**). For example, **g** gets replaced by **`7`**, **`z`** gets replaced by **`6`** (its place is 26, which has 6 as its last digit), etc. You may assume the string starts with a lower-case letter. Implementing this function in an efficient way is a significant part of the credit (hint: you want to implement this functionality without using if-statements, switch-statements or the ternary operator).

`cut`(): It takes in two strings and modifies the first string such that its length is now equal to that of the second string. If the first string is shorter than the second string, it does not get modified. As an extra constraint, you cannot declare any local variables inside this function (or inside any helper functions) other than the function parameters. If you decide to nevertheless use variables, you can, but you will lose half the points for this part of the question.

Your task: Implement these three functions **`next`**(), **`modify`**() and **`cut`**() below.

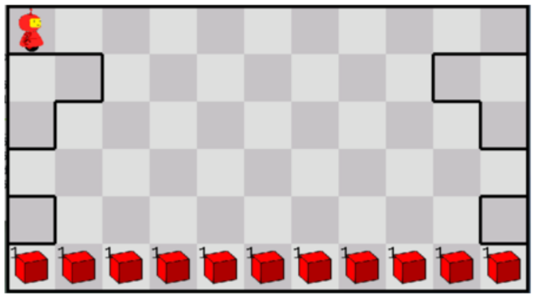
Question 5 (15 pts)

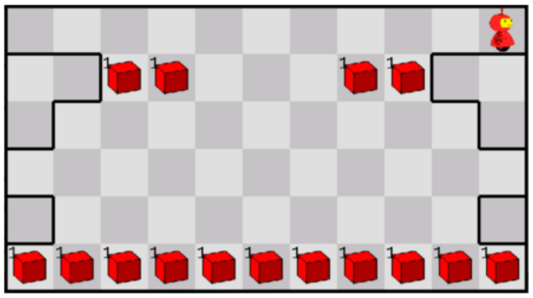
For this karel problem, **you are not allowed to use variables**. You will receive 0 credit if you use variables. All your functions need to have a void return type and not have any parameters. The repeat() functionality is available. This is similar to what you did in PA1 and PA1b.

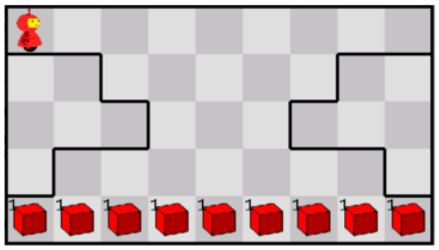
Karel finds himself in a map with symmetrical walls on the east and west side. The north-most row will always be completely open, i.e., have no walls and no items. The south-most row will have no walls and a single item on each square. Karel will start in the north-west corner, facing east, and with an infinite number of items in his bag. The dimensions of the map and the positions of the walls may vary. Other than the items in the south-most row, there are no other items present.

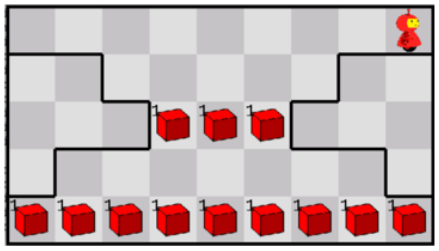
Karel must find the row with the narrowest span (i.e., the row with the shortest distance between the west and east walls) and deposit a single item on two consecutive squares on each side of the span. If the span is less than 4 squares wide, each square should have a single item. In other words, he should drop at most 4 items (2 on each side) and a square should have at most 1 item. Also look at the examples below. The span will always be at least one square wide (i.e., there will always be an open column). There will only be one narrowest span. Hint: because of the symmetry of the walls, the narrowest span is also where the west-walls extend the farthest to the east.

Karel needs to end on the north-east square, but his direction does not matter. No items should be removed from the map and no other items should be added other than the ones placed in the narrowest span as explained above.


Start state


End state


Start state


End state

Your task: Write this Karel program (on the next page). For your convenience, the code skeleton for a karel program is shown there and you may assume the correct settings and map files are available. It is important to use appropriate helper functions and top-down modular design. <u>Your code needs to be well-structured and easy to understand to receive credit</u>. <u>Code structure will account for a significant portion of the credit that can be earned on this question</u>.

```
#include <karel.h>

// Helper functions would go here

int main() {
   karel_setup("settings/settings01.json");

   // Your main code would go here
}
```

Write your complete karel program below (you can also use the back of the pages if you need more space, but make sure you clearly indicate where we can find your code). Make sure it is clear what does into your `main`(), but you do not need to copy the `karel_setup()` line.

---- END ----