SID: ________________

University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Fall 2017 Instructors: Randy Katz, Krste Asanovic 2017-09-26

😫 CS61C MIDTERM 1 😀

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	Perfect
First Name	Petra
Student ID Number	0xDEADBEEF
CS61C Login	cs61c-zzz
*The name of your **SECTION** TA and time*	
Name of the person to your LEFT	
Name of the person to your RIGHT	
All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. ***(please sign)***	

Instructions (Read Me!)

- This booklet contains 6 numbered pages including the cover page.
- Please turn off all cell phones, smartwatches, and other mobile devices. Remove all hats & headphones. Place your backpacks, laptops and jackets under your seat.
- You have 80 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use one handwritten 8.5"x11" page (front and back) crib sheet in addition to the RISC-V Green Sheet, which we will provide.
- There may be partial credit for incomplete answers; write as much of the solution as you can. **We will deduct points if your solution is far more complicated than necessary**. When we provide a blank, please fit your answer within the space provided.

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**Total**
Points Possible	12	19	20	19	20	90

SID: ________________

Q1: Back to the Base-ics (12 points)

a) Show how the binary string 0b1011 0110 can be interpreted and displayed as the following types:

Hexadecimal: 0x_____B6____

Unsigned Decimal: ______182_____

Two's Complement Decimal: ______-74_____

b) What is the minimum number of bits needed to represent all the unsigned integer values that a three-digit base-7 number could encode? Your answer should be a simplified decimal value.

Powers of 7 are shown below for reference:

7^1	7^2	7^3	7^4	7^5
7	49	343	2401	16807

______9_____

c) What bias should be added for a biased three-digit base-7 number to yield an equal number of positive and negative numbers? Your answer should be a simplified decimal value.

_____-171______

d) Convert the unsigned number 0xDF to its base-7 equivalent (i.e. the base-7 number with the same decimal value). What is the resulting number? The prefix 0s is for base-7.

0s______436_____

SID: ________________

Q2: Thanks for the Memories (19 points)

```
#define MAX_WORD_LEN 100
int num_words = 0;
void bar(char **dict) {
    char word2[] = "BEARS!";
    dict[num_words] = calloc(MAX_WORD_LEN, sizeof(char));
    strcpy(dict[num_words], word2);
    num_words += 1;
}
int main(int argc, char const *argv[]) {
    const int dict_size = 1000;
    char **dictionary = malloc(sizeof(char *) * dict_size);
    char *word1 = "GO";
    bar(dictionary);
    bar(dictionary);
    return 0;
}
```

Consider the program above. Based on what the given C expressions evaluate to, please select comparators to fill in the blanks (for 1-4) or the correct address type (for 5-7). As per the C standard, you cannot assume calls to `malloc` return heap addresses in a sequential order.

1. `&dictionary ___ &num_words`
   - ○ [✓ >]
   - ○ <
   - ○ ==
   - ○ Can't tell
2. `dictionary ___ &dict_size`
   - ○ >
   - ○ [✓ <]
   - ○ ==
   - ○ Can't tell
3. `&word1 ___ &dict`
   - ○ [✓ >]
   - ○ <
   - ○ ==
   - ○ Can't tell
4. `dictionary[1]___dictionary`
   - ○ >
   - ○ <
   - ○ ==
   - ○ [✓ Can't tell]
5. What type of address does `word1` evaluate to?
   - ○ Stack address
   - ○ Heap address
   - ○ [✓ Static address]
   - ○ Code address
6. What type of address does `&(word2[1])` evaluate to?
   - ○ [✓ Stack address]
   - ○ Heap address
   - ○ Static address
   - ○ Code address
7. What type of address does `*dictionary` evaluate to?
   - ○ Stack address
   - ○ [✓ Heap address]
   - ○ Static address
   - ○ Code address

Q3: Put it in Reverse (20 points)

1. Fill in the blanks to complete the `reverse` function which takes in a `head_ptr` to the head of a linked list and returns a **new copy** of the linked list in reverse order. You must allocate space for the new linked list that you return. An example program using reverse is also shown below.

```
struct list_node {
      int val;
      struct list_node* next;
};
struct list_node* reverse( ___struct list node**____ head_ptr ) {
      struct list_node* next = NULL;
      struct list_node* ret;
      while (*head_ptr != NULL) {
         ret = malloc(sizeof(struct list_node))______ ;
         ret->val = (*head_ptr)->val_________________ ;
         ret->next = next__________________________ ;
         next = ret__________________________________ ;
         *head_ptr = (*head_ptr)->next;
      }
      return ret_____________________________________;
}
/* Assume that NEW_LL_1234() properly malloc's a linked list
 * 1->2->3->4, and returns a pointer that points to the first
 * list_node in the linked list. Assume that before test_reverse
 * returns, head and ret will be properly freed. */
void test_reverse() {
      struct list_node* head = NEW_LL_1234();
      assert(head->val == 1); // returns True
      assert(head->next->val == 2); // returns True
      struct list_node* ret = reverse(&head);
      assert(head != ret); // ret is a new copy of the original list
      assert(ret->val == 4); // should return True
      . . .
}
```

2. If the function `test_reverse` is called, there will be one error. This error will result due to one of the lines already given to you in `reverse()`, from part 1 above. In five words or less, what is the error? There are no syntax-related errors.

_______memory leak_____________________

Q4: Ternary Search Tree Is Back (19 points)

Recall the Trie Tree and Ternary Search Tree from Homework #1. You've already implemented `memory_trie_node`, and now we ask you to provide the same feature for a Ternary Search Tree. Recall that the `TSTnode` structure needs to hold a `char` self, a `char*` word, and three `TSTnode` pointers to the left, right and sub trees.

1. First of all, please select all correct `TSTnode` structures from below. Please write your answer as letters in alphabetic order on the blank to the right:

   ______B______

A.
```
struct TSTnode {
    char self;
    char* word;
    TSTnode* left, right, sub;
};
```

B.
```
struct TSTnode {
    char self;
    char* word;
    struct TSTnode *left, *right, *sub;
};
```

C.
```
struct TSTnode {
    char* self;
    char* word;
    TSTnode *left, *right, *sub;
};
```

D.
```
struct TSTnode {
    char self;
    char *word;
    struct TSTnode* left, right, sub;
};
```

2. How many bytes does a single `TSTnode` from HW1 take up in memory? Assume that we are working on a **32 bit word-aligned architecture**, as we have normally in class.

   `sizeof(struct TSTnode)` = ___20________________

3. Assume you have the `TSTnode` struct, as defined in the project. Fill in `memory_tst_node` to calculate the total amount of heap memory usage (similar to what you did in `Trie` Tree). You may or may not need to use all blanks;

```
int memory_tst_node( struct TSTnode* node ) {
    if (!node)
        Return 0;
    unsigned int bytes = sizeof(struct TSTnode);
    bytes += memory_tst_node(node->left);
    bytes += memory_tst_node(node->right);
    bytes += memory_tst_node(node->sub);
    If (node->word)
        bytes += (strlen(node->word)+1)*sizeof(char);
    return bytes;
}
```

Q5: RISC-Y Business (20 points)

You wish to speed up one of your programs by implementing it directly in assembly. Your partner started translating the function is_substr() from C to RISC-V, but didn't finish. Please complete the translation by filling in the lines below with RISC-V assembly. The prologue and epilogue have been written correctly but are not shown.

Note: strlen(), both as a C function and RISC-V procedure, takes in one string as an argument and returns the length of the string (not including the null terminator).

```
/* Returns 1 if s2 is a substring of
s1, and 0 otherwise. */
int is_substr(char* s1, char* s2) {
  int len1 = strlen(s1);
  int len2 = strlen(s2);
  int offset = len1 - len2;
  while (offset >= 0){
    int i = 0;
    while (s1[i + offset] == s2[i]){
      i += 1
      if (s2[i] == '\0')
        return 1;
    }
    offset -= 1;
  }
  return 0;
}
```

```
1.  is_substr:
2.      mv s1, a0
3.      mv s2, a1
4.      jal ra, strlen
5.      mv s3, a0
6.      mv a0, s2
7.      jal ra, strlen
8.      sub s3, s3, a0
9.  Outer_Loop:
10.     __blt__  __s3__, _x0_, False
11.     add t0, x0, x0
12. Inner_Loop:
13.    add t1, t0, s3
14.    add t1, s1, t1
15.     lbu t1, 0(t1)
16.     ___add t2 s2 t0_____________
17.     ___lbu t2 0(t2)_____________
18.     _bne_ t1, _t2__, Update_Offset
19.     addi t0, t0, 1
20.     add t2, t0, s2
21.     _lbu t2 0(t2)______________
22.     beq t2, __x0___, ____True______
23.     jal x0 Inner_Loop
24. Update_Offset:
25.     addi s3, s3, -1
26.     __jal x0 Outer_Loop_________
27. False:
28.     xor a0, a0, __a0_____
29.     jal x0, End
30. True:
31.     addi a0, x0, 1
32. End: ....
```