University of New Mexico

CS 241: Data Organization using C
Final Exam

You may use one page of hand written notes (both sides) and a dictionary.
No i-phones, calculators nor any other device with transistors.

1) Linked List: This C program compiles and runs. It correctly adds char arrays (names) to a linked list in lexical order. What is its output?

syntax of `strcmp` from `string.h`

```
int strcmp ( const char * str1, const char * str2 )
```

This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.
Returns: zero if both strings are equal. A value greater than zero if the first character that does not match has a greater value in str1 than in str2. Otherwise, returns a value less than zero.

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

#define MAX_NAME_LEN 16

struct Node
{
  struct Node *next;
  char name[MAX_NAME_LEN];
};

struct Node *start = NULL;

void addNode(char* name);

//==========================================================
void main(void)
//==========================================================
{
  addNode("to");
  addNode("thine");
  addNode("own");
  addNode("self");
  addNode("be");
  addNode("true");
}

```

```c
//=============================================================
void addNode(char* name)
//=============================================================
{
  struct Node *newNode = malloc(sizeof(struct Node));
  strcpy(newNode->name, name);  //string copy

  if(start == NULL)
  { start = newNode;
    start->next = NULL;
    printf("addNode(%s) start=%s\n", name, start->name);
  }

  else if (strcmp(start->name, name) > 0)
  { newNode->next = start;
    start = newNode;
    printf("addNode(%s) start=%s\n", name, start->name);
  }
  else
  {
    printf("addNode(%s) ", name);
    struct Node *node = start;
    while(1)
    {
      if (node->next == NULL)
      { node->next = newNode;
        newNode->next = NULL;
        break;
      }

      else if (strcmp(node->next->name, name) > 0)
      { newNode->next = node->next;
        node->next = newNode;
        break;
      }
      node = node->next;
    }
    printf(" %s %s\n", node->name, node->next->name);
  }
}
```

2) Bit Operators: This C program compiles and runs. What is its output?

```
#include <stdio.h>
void main(void)
{ unsigned char x = 104;

  unsigned char a = x << 3;
  unsigned char b = x >> 3;
  unsigned char c = x & 13;
  unsigned char d = x & 60;
  unsigned char e = x | 24;
  unsigned char f = x ^ 24;;

  printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
          a, b, c, d, e, f);
}
```

3) This C program compiles and runs. If the output from lines 7 and 8 is:

```
sizeof(int)=4
x=0x7fff29af6538, x[0]=22
```

Then what is the output from line 10?

```
#include <stdio.h>

void main(void)
{
  int a[] = {22, 33, 44, 55, 66, 77};
  int *x = a;
  printf("sizeof(int)=%lu\n", sizeof(int));
  printf("x=%p, x[0]=%d\n", x, x[0]);
  x += 4;
  printf("x=%p, x[0]=%d\n", x, x[0]);
}
```

4) Substring: This C program compiles and runs. What is its output?

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

char *findSubstring(char *str, char *needle)
{
  int len = strlen(needle);
  int n = 0;

  while (*str)
  {
    printf("  %d %c%c\n", n, *str, *(needle+n)); //<<<<<<<<<<<<
    if ( *str == *(needle+n))
    {
      n++;
      if (n == len) return (str-len)+1;
    }
    else
    {
      str -= n;
      n = 0;
    }
    str++;
  }
  return NULL;
}


void main(void)
{
  char* sentence = "iHelHHello";
  char* word = "Hell";
  char* result = findSubstring(sentence, word);
  printf("sentence=%s\n", sentence);
  printf("result  =%s\n", result);                //<--
}
```

5) Binary Search: This C program compiles and runs. What is its output?

```
#include <stdio.h>

int binarySearch(int x, int v[], int high)
{
  int mid;
  int low = 0;

  while (low <=high)
  {
    mid = (low+high)/2;
    printf("[%d %d] ", low, high);

    if (x < v[mid]) high = mid-1;
    else if (x > v[mid]) low = mid+1;
    else return mid;
  }
  return -1;
}

void main(void)
{
  int nums[]={12, 33, 45, 47, 53, 55, 59, 73, 91, 93};
  int n =  sizeof(nums)/sizeof(int);
  printf("idx = %d\n", binarySearch(33, nums, n));
  printf("idx = %d\n", binarySearch(59, nums, n));
}
```