Software Development Engineer Interview Exercises

Exercise #1 – Programming in C or C++

Pick one exercise from this list. Make sure the program compiles, executes and has no memory leaks. The program should use stdin and stdout for reading and writing the answers. Please send the files with the solution to the chosen exercise.

1. Write a program that take a string as input and outputs this input string in reverse. For example if the input is “abcdef” the programs output should be “fedcba”
2. Write a program that takes a number of unsigned integers (max 100), and then prints every third integer in the list. Extra: create a function that randomize the unsigned integers.
3. Write a program that takes a string input, and can determine in the string is a palladium or not. “abba” is a palladium, “abcd” is not a palladium.

Exercise #2 – Programming in C

Program description

This program is a simple calculator that reads its input from a character buffer. If integers are read, they are pushed on a stack. If one of the operators (+ - * /) is read, the top two elements are popped off the stack, the operation is performed on them, and the result is pushed on the stack. The = operator writes out the value of the top element of the stack to a buffer.

Exercise

1. Assume the following code was sent by a peer for your review before doing merge with the master repository. What would be your observations regarding code optimization and errors? Include your inputs on the original file as comments.
2. Debug and update the program to display in the stdout any operation and the corresponding result. Example:

   10/2+8=13
3. Create new a function that accepts some integers and find the highest value. Example of the operation: `"2 18 24 4 5 H = 24"`

CALC.H

```
/*----- FILE CALC.H ----------------------------------------------------*/
/*                                                                      */
/* Header file for CALC.C PUSHPOP.C READTOKN.C                          */
/* a simple calculator                                                  */
/*----------------------------------------------------------------------*/
typedef enum toks {
  T_INTEGER,
  T_PLUS,
  T_TIMES,
  T_MINUS,
  T_DIVIDE,
  T_EQUALS,
  T_STOP
} Token;
Token read_token(char buf[]);
typedef struct int_link  {
  struct int_link * next;
  int i;
} IntLink;
typedef struct int stack {
  IntLink * top;
} IntStack;
extern void  push(IntStack *, int);
extern int pop(IntStack *);
```

CALC.C

```
/*----- FILE CALC.C ----------------------------------------------------*/
/*                                                                      */
/* A simple calculator that does operations on integers that            */
/* are pushed and popped on a stack                                     */
/*----------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include "calc.h"
IntStack stack = { 0 };
main()
{
  Token tok;
  char word[100];
  char buf_out[100];
  int num, num2;
  for(;;)
  {
    tok=read_token(word);
    switch(tok)
    {
      case T_STOP:
        break;
      case T_INTEGER:
        num = atoi(word);
        push(&stack,num);     /* CALC1 statement */
        break;
      case T_PLUS:
        push(&stack, pop(&stack)+pop(&stack) );
```

```
        break;
      case T_MINUS:
        num = pop(&stack);
        push(&stack, num-pop(&stack));
        break;
      case T_TIMES:
        push(&stack, pop(&stack)*pop(&stack));
        break;
      case T_DIVIDE:
        num2 = pop(&stack);
        num = pop(&stack);
        push(&stack, num/num2);    /* CALC2 statement */
        break;
      case T_EQUALS:
        num = pop(&stack);
        sprintf(buf_out,"= %d ",num);
        push(&stack,num);
        break;
    }
    if (tok==T_STOP)
      break;
  }
  return 0;
}
```

PUSHPOP.C

```
/*----- FILE PUSHPOP.C -------------------------------------------------*/
/*                                                                      */
/* A push and pop function for a stack of integers                      */
/*----------------------------------------------------------------------*/
#include <stdlib.h>
#include "calc.h"
/*----------------------------------------------------------------------*/
/* input:  stk - stack of integers                                      */
/*         num - value to push on the stack                             */
/* action: get a link to hold the pushed value, push link on stack      */
/*                                                                      */
extern void push(IntStack * stk, int num)
{
  IntLink * ptr;
  ptr       = (IntLink *) malloc( sizeof(IntLink));  /* PUSHPOP1   */
  ptr->i    = num;                /* PUSHPOP2 statement */
  ptr->next = stk->top;
  stk->top  = ptr;

}
/*----------------------------------------------------------------------*/
/* return: int value popped from stack                                  */
/* action: pops top element from stack and gets return value from it    */
/*----------------------------------------------------------------------*/
extern int pop(IntStack * stk)
{
  IntLink * ptr;
  int num;
  ptr      = stk->top;
  num      = ptr->i;
```

```
  stk->top = ptr->next;
  free(ptr);
  return num;
}
```

READTOKN.C

```
/*----- FILE READTOKN.C -----------------------------------------------*/
/*                                                                     */
/* A function to read input and tokenize it for a simple calculator    */
/*---------------------------------------------------------------------*/
#include <ctype.h>
#include <stdio.h>
#include "calc.h"
/*---------------------------------------------------------------------*/
/* action: get next input char, update index for next call             */
/* return: next input char                                             */
/*---------------------------------------------------------------------*/
static char nextchar(void)
{
/*---------------------------------------------------------------------*/
/* input  action:                                                      */
/*    2      push 2 on stack                                           */
/*    18     push 18                                                   */
/*    +      pop 2, pop 18, add, push result (20)                      */
/*    =      output value on the top of the stack (20)                 */
/*    5      push 5                                                    */
/*    /      pop 5, pop 20, divide, push result (4)                    */
/*    =      output value on the top of the stack (4)                  */
/*---------------------------------------------------------------------*/
  char * buf_in  = "2 18 + = 5 / = ";
  static int index;     /* starts at 0 */
  char ret;
  ret = buf_in[index];
  ++index;
  return ret;
}
/*---------------------------------------------------------------------*/
/* output: buf - null terminated token                                 */
/* return: token type                                                  */
/* action: reads chars through nextchar() and tokenizes them           */
/*---------------------------------------------------------------------*/
Token read_token(char buf[])
{
  int i;
  char c;
  /* skip leading white space */
  for( c=nextchar();
       isspace(c);
       c=nextchar())
    ;
  buf[0] = c;  /* get ready to return single char e.g."+" */
  buf[1] = 0;
  switch(c)
  {
    case '+' : return T_PLUS;
    case '-' : return T_MINUS;
```

```
    case '*' : return T_TIMES;
    case '/' : return T_DIVIDE;
    case '=' : return T_EQUALS;
    default:
      i = 0;
      while (isdigit(c)) {
        buf[i++] = c;
        c = nextchar();
      }
      buf[i] = 0;
      if (i==0)
        return T_STOP;
      else
        return T_INTEGER;
  }
}
```