Real Time College

מרכז להכשרות מקצועיות והשמה בתעשיית ההייטק

Test - C for Embedded Systems

The following test is a simulation of questions used in a Professional Interview for a Real-Time \Embedded Engineer position; the questions will focus on the C language (Not RT Concepts).

The purpose of this test is to help you prepare for the Professional Interviews and determine the level of your knowledge in C for embedded systems.

Using the test we can find out more accurately which subjects you will need to strengthen and achieve the goal of preparing you to master the Science of RT\Embedded Engineering.

The subjects you will be questioned on are:

1. C for embedded systems

The time to complete the test is 2 hours

Good luck

Question 1

Implement Stack Using the following data member & Functions (Already implemented & defined for you):

```
typedef struct Node
{
     int data;
     struct Node * next;
}  Node;
```

1. **Node* new_node(int value)**
Creates new allocation for new Node , initialize it , and returns it.
2. **Node* add_node(Node *head, Node *new_node)**
Enter the Node that new_node is pointing to, to the right location in the list, so that the list remains arranged from the small element to big element (assuming it was already arranged in that order). The function returns pointer to the head of the list.
3. **void free_list(Node *head)**
free the whole list.

Stack is the data structure which operates on LIFO (Last In First Out).
Meaning , the last element which enters the list , will be the first to go out.

You need to implement stack using the following data structure:

```
typedef struct stack
{
  Node *head;
  int size;
} Stack;
```

Implement the following functions:

1. **void push(Stack *stk, int value);**
   Push – enters a new element to the head of the list.

2. **int pop(Stack *stk);**
   pop – takes out the first element at the top of the list , and returns its value.
   Make sure to check stack isn't empty.

3. **int is_empty(Stack *stk);**
   is_empty returns true id stack is empty , and false otherwise.

4. **Stack* new_stack(void);**
   New_stack allocates new stack struct and initiate it.

5. **void free_stack(Stack *stk);**
   free_stack – release the stack and all the elements inside.

Question 2:

What is the output of the following programs (it might not compile at all… pay attention)?

a.

```
#include<stdio.h>

struct xx
{
int x=3;
char name[]="hello";
};

main()
{

struct xx *s;
        printf("%d",s->x);
        printf("%s",s->name);

}
```

Output:

b.

```
void swap(double *a, double *b);

int main()
{
        int *a, *b;
        *a=4;
        *b=5;
        swap(a,b);

}

void swap(int *a, int *b)
{
        int tmp;

        tmp = *a;
        *a = *b;
        *b = tmp;
}
```

Output:

c.

```
        #include <stdio.h>
        int *call();
        int *call2();

        void main()
        {
          int *ptr,*ptr2;
          ptr=call();
          ptr2=call2();

         printf("%d",*ptr);
        }

        int * call()
        {
                int x=20;
                ++x;
                return &x;
        }
        int * call2()
        {
                int x=30;
                ++x;
                return &x;
}
```

Output:

:

Question 3

A Network system reads packets from the device driver's buffer.

- All packets are fixed in size.
- Each packet includes 32 bit Header and 1k Data fields .
- Header fields:
  SOF –Start of Frame 8 bit.
  SID - Subscriber I.D 12 bit
  CC- Continuity Check 4 bit.

The Packets are part of a streamed VOD and have to be synchronized per SID, to ensure that the CC field is used. The CC field should increment by 1 for 2 consecutive packets per SID.

SOF	
SID	
SID	not used
not used	CC
1K DATA	

1. Write the appropriate structure for each packet.

2. Write a function which verifies the Continuity Check, the function should recognize missing packets between 2 consecutive packets with the same SID and store the number of missing packets per SID.

3. Point out the system's limitations.