Sistemas Digitais II
1ª Repetição de 1998/99

Leia atentamente o enunciado. Seja breve nas respostas, mas justifique-as convenientemente. Por favor, use letra legível ! Quando tiver que escrever programas, deverá apresentar uma listagem com comentários que facilitem a compreensão do programa. Tem 90 min para completar a repetição. Boa sorte !

1) Um determinado programador (com intuitos duvidosos, e um estilo de programação que me faz lembrar os trabalhos práticos …), escreveu o seguinte programa:

```
#include <stdio.h>
#include "mydefs.h"/*este ficheiro*/
char tambriel;     /* esta vazio */
int i=0, j=1;
int Func1(int a);
int Func2(int *a);
int Func3(int a);
main( )
{
  int i=2,j=3,*k;
  char fraivens;
  printf("Starting main\n");
  k=&j;
  printf("main->i=%d\n",i);
  j++;
  i=Func1( i );
  printf("main->i=%d\n",i);
  printf("main->k=%d\n",*k);
  j=Func2( k );
  printf("main->j=%d\n",j);
  i=Func3(*k);
  printf("main->i=%d\n",i);
  printf("main->k=%d\n",*k);
  printf("\nEnd of silly prog\n");
  return 0;
}
int Func1(int a)
{
 int i=10;
 int corjoby;
 printf("Starting Func1\n");
 a=a+i;
 printf("func1->a=%d\n",a);
 for( i-- ; i>0; i--)
   printf("%d,",i);
 printf("\n");
 printf("Finished Func1\n");
 return a-9;
}
int Func2(int *a)
{
 int j=20;
 printf("Starting Func2\n");
 printf("Func2->i=%d\n",i);
 printf("Func2->a=%d\n",*a);
 switch( *a )
  {
   case 3: printf("X");
   case 4: printf("Y");
   case 5: printf("Z\n"); break;
   default: printf("MUU\n");
  }
 Func1( j );
 printf("Finished Func2!\n");
 return 2;
}

int Func3(int a)
{
 int i;
 printf("Starting Func3\n");
 i=a;
 if( i>0 )
   Func3(a-1);
 else
   printf("Func3->Ending...\n");

 printf("Func3->i=%d\n",i);
 printf("Finished Func3\n");
 return i+1;
}
```

1.1) Qual a diferença entre os "includes" da primeira e da segunda linha do programa?
1.2) A função Func3 pode usar a variável tambriel ? E a variável corjoby ? E a fraivens ? Porquê ?
1.3) Qual o output para o ecrã deste programa ?

2) Com as perícias adquiridas em C e a prática do PCRobots, pode melhorar a peça *"G. Andachtrondo"* que andou a desenvolver o ano passado. Por hipótese, conseguiu-se um compilador de C para o 8085, e alguém já escreveu as seguintes rotinas:

int GetNextTarget(int *azi, int *elev) - Devolve o azimute e elevação do próximo alvo seleccionado pelo sistema de busca. O valor de rotorno é 1 se houver algum alvo, e NULL se não houver nenhum.
int GetElev() – Devolve a elevação actual da peça
int GetAzi() – Devolve o Azimute actual da peça
void GoUp() – Actua o motor que faz subir a elevação da peça
void GoDown() – Actua o motor que faz descer a peça
void GoMore() – Actua o motor que aumenta o azimute (0-360) da peça

void GoLess() – Acuta o motor que diminui o azimute (0-360) da peça
int Stop() – Pára os motores que fazem mover a peça
int GunStable() – Devolve 1 se a peça estiver estável, e 0 se estiver ainda a vibrar.
void Wait() – Espera por breves momentos, permitindo que o CPU faça outras tarefas.
void fire() – Dispara a peça.
int GetNozzleSpeed() – Devolve a velocidade à boca (em m/s)
int GetTemp() – Devolve a tempratura da peça (em ºC)
void PrintAlarm(char code) – Acende um LED de alarme, e envia para um display de 7 segmentos o código de alarme (que pode ser A,L,C, ou 0)

2.1) Escreva um programa com um loop infinito que vai pedindo alvos (eventualmente esperando que estejam disponíveis), aponta a peça, espera que esta estabilize, dispara, e repete o processo. Se a temperatura após o disparo subir acima de 500ºC ou a velocidade à boca fôr inferior a 300m/s, deverá enviar o sinal de alarme com os códigos C (de Calor) ou L (de Lento).
NOTA: se quiser ter cotação máxima, não obrigue a peça a rodar mais que 180º em azimute, ou seja, se ela está apontada para 010 e tem que disparar para 350, deverá andar 20º no sentido inverso dos ponteiros do relógio, e não 340 no sentido directo.

2.2) A rotina PrintAlarm teve alguns problemas, e por isso terá que a escrever novamente (usando sempre C). O display de 7 segmentos está num porto de I/O mapeado para memória para o endereço 0199. O bit menos significativo é o segmento "a", o seguinte o "b", e assim sucessivamente, o bit mais significativo é usado para o ponto decimal, que funciona também como LED de alarme. Escreva a rotina PrintAlarme.

3) Escreva um conjunto de rotinas para implementar um stack em software para guardar caracteres. O sistema deverá usar variáveis dinâmicas de modo a só gastar mais espaço de memória quando este é mesmo necessário. Deverá escrever as seguintes rotinas:

i) **stack *InitStack()** — Inicializa o stack, devolvendo um apontador para o mesmo.

ii) **int Push(stack *this_stack, char value)** — Põe um dado no stack apontado por "stack". Devolve 1 se tudo correu bem, e –1 se houve problemas ao allocar memória.

iii) **char Pop(stack *this_stack)** — Devolve um dado do stack "stack". Devolve NULL se já não há dados válidos no stack.

iv) **int GetSize(stack *this_stack)** — Devolve o número de dados que está no stack.

Deverá ainda definir o tipo "stack" (e outros tipos que eventualmente utilize), e fazer um esboço que explique a estrutura to stack. Finalmente esctrva um programa que use essas rotinas. Esse programa deverá criar um stack, guardar lá os caraceres O,L,A, verificar o tamenho do stack, e tentar retirar 4 valores do stack. (comentário: se preferir testar resolvendo o problema das Torres de Hanoi com 3 stackes destes, pode fazê-lo…no, just joking…)

Sugestão: Quando inicializar o stack, crie uma ficha em branco, que não é usada para guardar dados, mas apenas para "marcador" e "ponto de partida". Óbviamente que o Stack será uma lista, com cada push a allocar mais memória, e cada pop a libertá-la.