MAC2166 – Introdução à Computação para Engenharia

Escola Politécnica

Segunda – 16 de maio de 2011

Nome: _____

Assinatura: _____

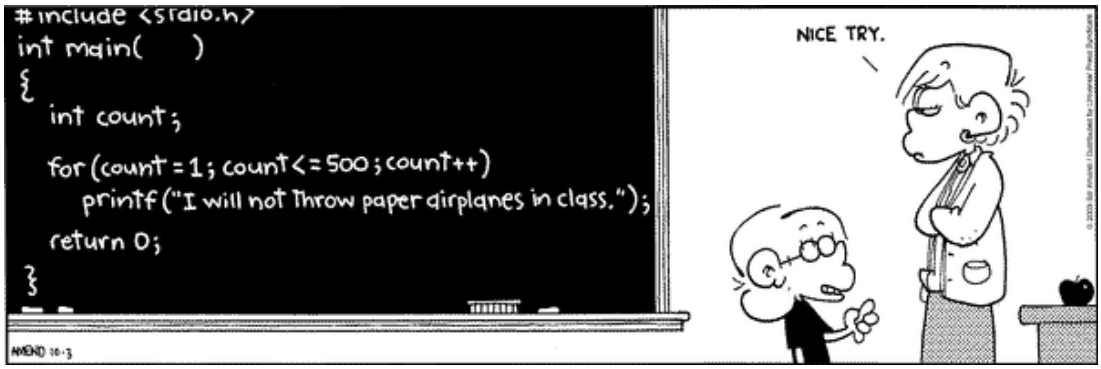
Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. Qualquer questão pode ser resolvida em qualquer página. Se a questão não está na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO X em letras ENORMES antes da solução.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é permitido o uso de calculadoras.
7. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas



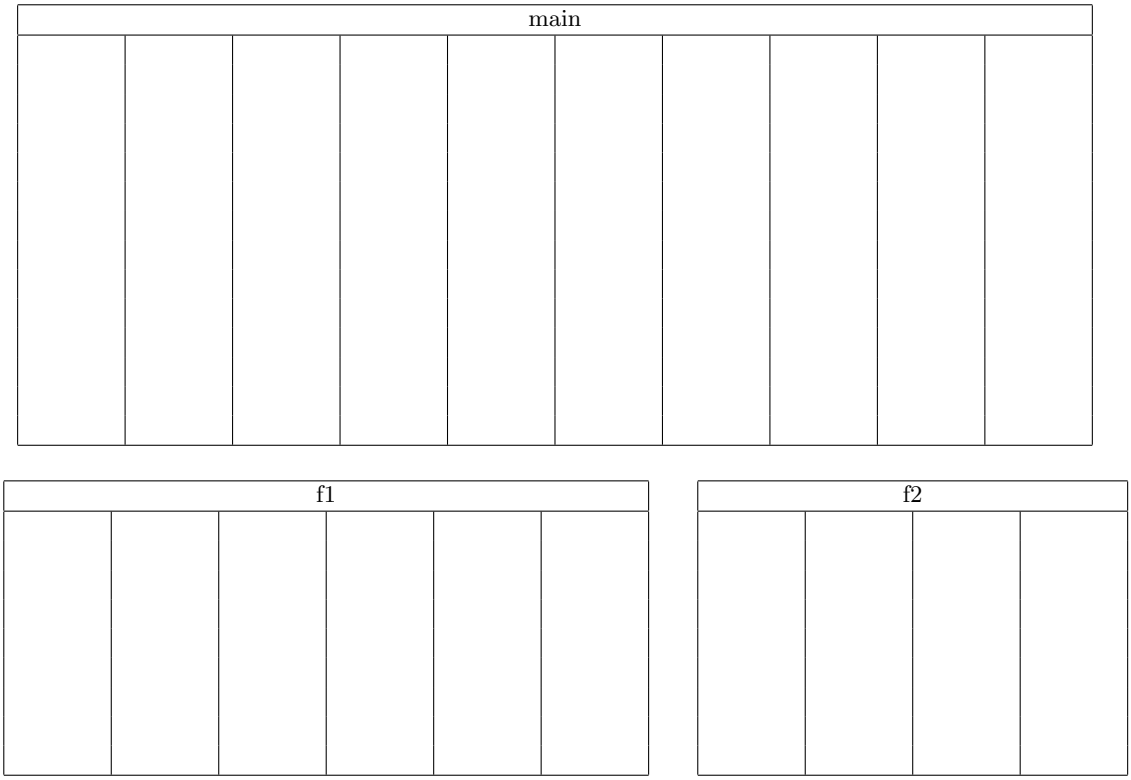
Questão	Valor	Nota
1	2	
2(a)	1	
2(b)	1.5	
2(c)	1	
2(d)	0.5	
3(a)	1	
3(b)	3	
Total	10,0	

Questão 1 (vale 2 pontos)
Simule a execução do programa abaixo, destacando a sua saída. A saída do programa consiste de tudo que resulta dos comandos `printf`. **Para efeito de correção só será considerada a saída do programa**.

```
#include <stdio.h>

float f1 (int a, int b, int *c);

int f2 (int a, float *b);

int main () {
   int nusp, a, b, c;
   float x;

   printf ("Digite o seu no. USP: ");
   scanf ("%d", &nusp);
   printf ("nusp=%d\n", nusp);

   a = (nusp%5)-7;
   b = 1 + a;
   c = (2*b+1)/2;
   printf("1: a=%d b=%d c=%d\n", a,b,c);
   x = f1(b,c,&a);
   printf("3: a=%d b=%d c=%d x=%f\n", a,b,c,x);

   a = (nusp%5)-7;
   x = 1.5;
   b = f2(a,&x);
   printf("5: a=%d b=%d x=%f\n", a,b,x);

   return 0;
}
```

```
float f1 (int a, int b, int *c) {
   float x;
   int i,j;
   a = (a+b*(*c))%10;
   *c = a-b;
   b = b*b;
   i = *c;
   j = i+(*c);
   x = ((float)b+(*c)+i+j)/10;
   printf("2: i=%d j=%d x=%f\n",i,j,x);
   return x;
}

int f2 (int a, float *b) {
   float x;
   int c;
   x = (int) (*b);
   c = (x+a)/10;
   x = x+a/10;
   *b = (*b)*(*b);
   printf("4: c=%d x=%f\n", c, x);
   return c;
}
```

Você pode usar a tabela abaixo como bem entender. Cada turma está habituada a simular de maneira diferente, fazendo tabelas com "caras" diferentes da abaixo.



Saída do programa

Questão 2 (vale 4 pontos)
item (2a) (vale 1 ponto)

Escreva uma função de protótipo:

```
int ehprimo (int i);
```

que recebe um inteiro $i \geq 1$ e retorna 1 se i for primo e 0 caso contrário.

item (2b) (vale 1,5 pontos)

Dois números primos p e q são *gêmeos* se $q = p + 2$. Por exemplo, 3 e 5 são números primos gêmeos e 101 e 103 também são.

Escreva uma função de protótipo:

```
void devolveprimosgemeos (int i, int *p1, int *p2);
```

que utiliza a função `ehprimo` do item **(2a)** e devolve em `*p1` e `*p2` o `i`-ésimo par de primos gêmeos, onde (3, 5) é o primeiro par de primos gêmeos, (5, 7) é o segundo, (11, 13) é o terceiro, ...

A sua função deve usar **obrigatoriamente** a função `ehprimo`. Você pode usar essa função **mesmo sem a ter escrito**.

item (2c) (vale 1 ponto)

A constante de Brun é calculada através da soma dos inversos dos números primos gêmeos. Portanto, a constante de Brun pode ser escrita como

$$\left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\ldots.$$

Escreva uma função de protótipo:

```
float calculaconstantebrun (int n);
```

que utiliza a função `devolveprimosgemeos` do item **(2b)** e retorna a soma dos inversos dos `n` primeiros pares de primos gêmeos da constante de Brun.

A sua função deve usar **obrigatoriamente** a função `devolveprimosgemeos`. Você pode usar essa função **mesmo sem a ter escrito**.

item (2d) (vale 0,5 ponto)

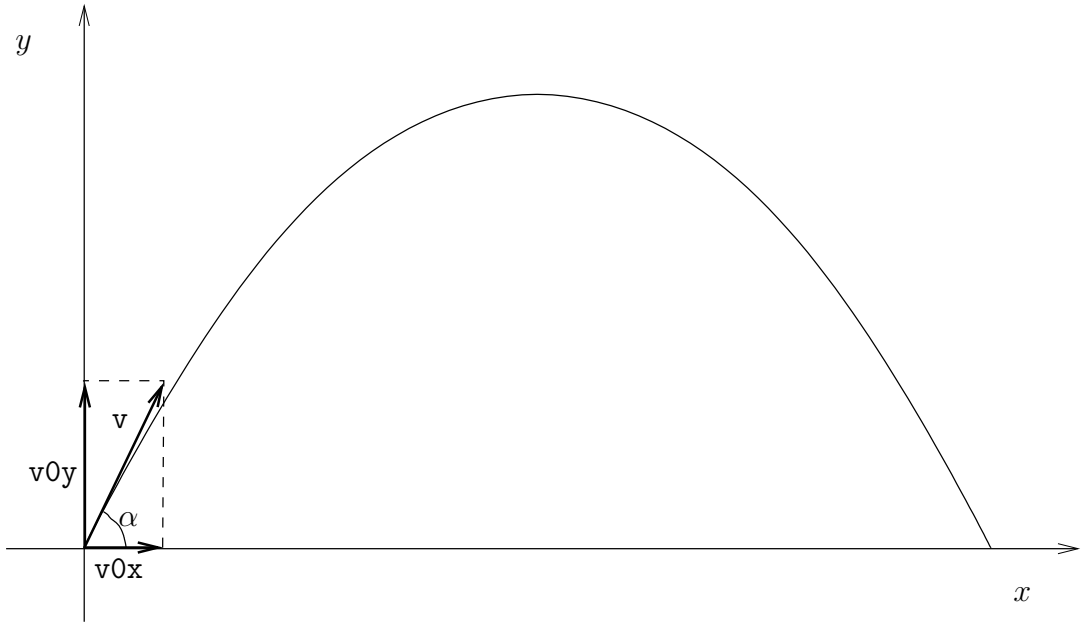
Escreva um programa que lê um inteiro positivo `n` e imprime a soma dos inversos dos `n` primeiros pares de primos gêmeos da constante de Brun, utilizando a função `calculaconstantebrun` do item **(2c)**.

Não há necessidade de copiar as funções dos itens anteriores aqui. Escreva somente a função `main` e eventualmente algum `#define` que porventura você esteja utilizando.

O seu programa deve usar **obrigatoriamente** a função `calculaconstantebrun`. Você pode usar essa função **mesmo sem a ter escrito**.

Questão 3 (vale 4 pontos)

Esta questão consiste na implementação de duas funções. Elas são simplificações **muito grandes** de alguns trechos de código que você escreveu para o seu EP3.



O arremesso de um projétil pode ser modelado considerando-se somente a força gravitacional agindo sobre ele. Isso implica em um **movimento uniforme** na horizontal e um **movimento uniformemente acelerado** na vertical.

Nesta questão, o objetivo é simular o movimento de um projétil com a modelagem acima. Vamos considerar que o projétil, no lançamento, se encontra na posição $(x_0, y_0) = (0, 0)$, onde o eixo x é horizontal e o eixo y é vertical.

Nesta questão é usada a seguinte declaração.

```
/* Aceleracao da gravidade na superficie da terra: */
#define G0   9.807
```

Considere que **nas fórmulas dadas** não há necessidade de nenhuma conversão, que todas as medidas são consistentes com as fórmulas. Somente há necessidade de conversão para o ângulo, que é dado em graus e deve ser convertido para radianos.

Você pode escrever mais funções se achar conveniente e pode usar qualquer função da biblioteca `math.h` que julgar necessário.

item (3a) (vale 1 ponto)
Escreva uma função de protótipo

```
void atualiza(float *vx, float *vy,  float *xP, float *yP, float deltaT);
```

que recebe a velocidade corrente `(*vx, *vy)` de um projétil, sua posição corrente `(*xP, *yP)` e um intervalo de tempo `deltaT`. A função devolve a nova velocidade do projétil em `*vx` e `*vy` e sua nova posição em `*xP` e `*yP` após o intervalo de tempo `deltaT`. Para tanto, use as seguintes equações:

$$\texttt{xP}_N = *\texttt{xP} + *\texttt{vx} * \texttt{deltaT},$$

$$\texttt{yP}_N = *\texttt{yP} + *\texttt{vy} * \texttt{deltaT} - \texttt{G0} * \texttt{deltaT}^2/2$$

e

$$\texttt{vy}_N = *\texttt{vy} - \texttt{G0} * \texttt{deltaT},$$

onde $(\texttt{xP}_N, \texttt{yP}_N)$ é a nova posição do projétil e $\texttt{vy}_N$ sua nova velocidade vertical. Note que a velocidade horizontal não se altera neste modelo.

Para utilizar a função `main` pedida no item **(3b)**, você deve usar as duas funções descritas a seguir **sem escrevê-las**, supondo que lhe foram **dadas**.

1. `int mostre_mapa(float xP, float yP, int *iP, int *jP);`

   Essa função recebe as coordenadas cartesianas `(xP,yP)` de um projétil e os índices `(*iP,*jP)` da região anterior do projétil no mapa. Ela calcula os índices da região correspondente ao ponto `(xP,yP)`. Se esses índices forem diferentes dos armazenados em `(*iP,*jP)`, então mostra o mapa com o projétil na nova posição no mapa e armazena em `(*iP,*jP)` os novos índices calculados. Retorna 1 caso o mapa tenha sido mostrado e 0 em caso contrário. Você pode supor que a posição `(0,0)` está na região de índices `(0,0)`.

2. `void mostre_dados(float t, float xP,  float yP, float vx, float vy, float v);`

   Essa função recebe o tempo `t` de simulação, a posição atual `(xP,yP)`, a velocidade atual `(vx,vy)` e a intensidade da velocidade atual do projétil, `v`. A função imprime estes dados.

No item **(3b)** você deve usar **obrigatoriamente** as funções `atualiza`, `mostre_mapa` e `mostre_dados`. Você pode usar essas funções **mesmo sem tê-las escrito**.

item (3b) (vale 3 pontos)
Escreva uma função `main` que leia

1. o tempo máximo de simulação e o intervalo `deltaT` entre um instante e o instante seguinte da simulação, ambos em segundos;

2. o ângulo α de lançamento, em graus, do projétil;

3. a intensidade `v` da velocidade de lançamento de um projétil;

Todos esses dados são do tipo real (`float`).

O programa deve calcular a trajetória do projétil, sujeito à força gravitacional, começando na posição `(0,0)` com velocidade `(v0x,v0y)`, que é calculada a partir dos dados da intensidade da velocidade e do ângulo.

O instante inicial da simulação é 0 e o instante seguinte a um dado instante `t` é o instante `t + deltaT`.

A simulação da trajetória deverá terminar quando o tempo de simulação ultrapassar o tempo máximo de simulação ou quando o projétil atingir o chão. O chão tem coordenada y igual a zero.

Um mapa com a posição do projétil deve ser mostrado na tela no início e durante a simulação. Durante a simulação, o mapa deve ser mostrado somente quando o projétil muda de região. Sempre que o mapa for impresso, a função `mostre_dados` deve ser chamada para imprimir os dados atuais do projétil.

O programa deve também imprimir, ao final da simulação, uma mensagem indicando se o projétil atingiu o chão ou o tempo de simulação terminou.

Não há necessidade de copiar as funções dos itens anteriores aqui. Escreva somente a função `main` e eventualmente algum `#define` que porventura você esteja utilizando. Você pode usar a função `atualiza` mesmo sem a ter escrito.