MAC 2166 – Introdução à Computação

Engenharia Elétrica e de Computação

Primeiro Semestre de 2023

Prova de Recuperação – 21 de julho de 2023

Nome completo: _____

NUSP: _____ Turma: _____

Assinatura: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. Preencha o cabeçalho acima.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é permitido o uso de artefatos eletrônicos.
7. Não é permitida a consulta a livros, apontamentos ou colegas.
8. Não é necessário apagar rascunhos no caderno de questões.

DURAÇÃO DA PROVA: 2 horas

Questão	Nota
1	
2	
3	
Total	

Q1 (3.0 pontos) Seja v um vetor com N inteiros: $v_0, v_1, \ldots, v_{N-1}$. Para cada índice i ($0 \leq i < N$), definimos h_i como sendo o maior h tal que $v_i, v_{i+1}, \ldots, v_{i+h-1}$ têm todos a mesma paridade. Definimos w como sendo o vetor com as entradas h_i ($0 \leq i < N$). Note que w é um vetor com N entradas.

Exemplo. Se $N = 15$ e

$$v = \Big[\ 3\ \ 1\ \ 4\ \ 1\ \boxed{5\ \ 9}\ \ 2\ \ 7\ \ 1\ \ 8\ \boxed{4\ \ 6\ \ 4}\ \ 1\ \ 5\ \Big], \tag{1}$$

então

$$w = \Big[\ 2\ \ 1\ \ 1\ \ 3\ \ 2\ \ 1\ \ 1\ \ 2\ \ 1\ \ 4\ \ 3\ \ 2\ \ 1\ \ 2\ \ 1\ \Big]. \tag{2}$$

Por exemplo, temos acima que $w[4] = h_4 = 2$, por conta dos 2 números ímpares marcados no vetor v em (1). Temos também que $w[10] = h_{10} = 3$, por conta dos 3 números pares marcados no vetor v em (1).

Nesta questão, você deve escrever um programa que recebe um vetor v com N inteiros e que tem como saída o vetor w associado. Além disso, seu programa deve imprimir o valor máximo em w e deve imprimir um índice j de w onde ocorre esse máximo. Havendo "empates", seu programa pode ter como saída qualquer j válido.

Parte do programa é dado. Sua tarefa é completar este programa.

```
#include <stdio.h>

#define NMAX 100

int read_array(int v[]);
void print_array(int w[], int N);
int same_parity(int w[], int v[], int N, int *i);
int same_parity_at_entry(int v[], int N, int i);

int main() {
  int N, sp, i;
  int v[NMAX], w[NMAX];

  N = read_array(v);
  sp = same_parity(w, v, N, &i);

  print_array(w, N);
  printf("Max same parity: %d (at %d)\n", sp, i);
  return 0;
}

int read_array(int v[]) {
  int N, i;
  scanf("%d", &N);
  for (i = 0; i < N; i++)
    scanf("%d", &v[i]);
  return N;
}
```

```
void print_array(int w[], int N) {
  int i;

  for (i = 0; i < N; i++)
    printf("%d ", w[i]);

  printf("\n");
}

/* Demais funcoes: a serem implementadas */
```

(a) Escreva uma função de protótipo

`int same_parity_at_entry(int v[], int N, int i);`

que recebe um vetor `v` de `N` inteiros e um índice `i` especificando uma entrada de `v`, e que devolve o valor h_i associado, como definido no começo dessa questão. Por exemplo, a chamada `same_parity_at_entry(v, 15, 4)`, com o vetor `v` dado em (1), deve devolver o valor 2.

(b) Escreva uma função de protótipo

```
int same_parity(int w[], int v[], int N, int *i);
```

que recebe um vetor `v` de `N` inteiros, e que armazena em `w` o vetor w definido no começo desta questão. Ademais, `same_parity()` deve devolver o valor máximo `max` presente em `w` e deve armazenar em `*i` um valor j tais que `w`$[j]$ = `max`. Sua função deve usar a função `same_parity_at_entry()` do item anterior, mesmo que você não a tenha escrito.

Q2 (4.0 pontos) Os assim chamados **números de Bell** B_N ($N \geq 0$) satisfazem a seguinte recorrência: $B_0 = 1$ e, para todo $N > 0$,

$$B_N = \sum_{0 \leq k < N} \binom{N-1}{k} B_k. \tag{3}$$

Assim, por exemplo, $B_1 = 1$, $B_2 = 2$, $B_3 = 5$, $B_4 = 15$ e $B_5 = 52$. Nesta questão, você deve escrever um programa que, dado $N \geq 0$, lista os números de Bell $B_0, B_1, \ldots, B_N$.

Observação. Não é relevante para a resolução desta questão, mas B_N é o número de partições que um conjunto de N elementos admite.

(a) Observe que coeficientes binomiais ocorrem em (3). Mais à frente, você escreverá uma função que preenche uma matriz com coeficientes binomiais. Para tanto, você deverá obrigatoriamente usar os seguintes fatos sobre coeficientes binomiais:

(A) para todo $i \geq 0$ temos que $\binom{i}{0} = 1$,

(B) para todo $j > 0$ temos que $\binom{0}{j} = 0$, e

(C) para todo $i > 0$ e $j > 0$, temos que

$$\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}. \tag{4}$$

Como aquecimento, termine o preenchimento da tabela abaixo, inserindo na linha i e coluna j o coeficiente binomial $\binom{i}{j}$ usando o fato (C) acima.

	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1					
2	1					
3	1					
4	1					
5	1					

(b) Escreva uma função de protótipo

```
void pascal(int P[][NMAX], int N);
```

que recebe uma matriz `P` e um natural `N`, e que preenche `P` com as primeiras `N + 1` linhas do triângulo de Pascal: após a execução de sua função, a entrada `P[i][j]` de `P` deve ser o coeficiente binomial $\binom{\texttt{i}}{\texttt{j}}$, para todo $0 \leq \texttt{i} \leq \texttt{N}$ e todo $0 \leq \texttt{j} \leq \texttt{N}$. Sua função deve obrigatoriamente usar os fatos (A), (B) e (C) do item (a) acima.

Exemplo. Para $\texttt{N} = 5$, sua função deve preencher a matriz `P` com os 36 números na tabela do item (a) que você preencheu.

(c) Escreva uma função de protótipo

```
void compute_bell(int bell[], int N);
```

que recebe um vetor `bell` e um natural `N` e que preenche `bell` com os primeiros `N + 1` números de Bell. Isto é, após a execução de sua função, deve valer que `bell`$[n] = B_n$ para todo $0 \leq n \leq N$. Sua função `compute_bell()` deve usar a função `pascal()` do item anterior, mesmo que você não a tenha escrito.

(d) Complete a escrita de seu programa: escreva uma função `main()` que lê N e que imprime os números de Bell $B_0, B_1, \ldots, B_N$. Por exemplo, com entrada 5, seu programa deve ter saída

```
0 1
1 1
2 2
3 5
4 15
5 52
```

Q3 (3.0 pontos) Esta questão propõe uma extensão de seu EP2, que simula um sistema de quatro blocos que colidem de forma elástica. Lembre que, para sua simulação, dada uma configuração do sistema, era necessário descobrir qual será a próxima colisão e em quanto tempo ela ocorrerá. Esta questão considera esse problema, mas em um sistema com N blocos. Lembre que os blocos têm dimensão $L \times L \times L = 0.1 \times 0.1 \times 0.1$.

Parte do programa é dado (veja abaixo). Sua tarefa é completar este programa, escrevendo as funções que faltam. No programa abaixo, o vetor `x[]` é usado para armazenar as posições dos centros dos N blocos e o vetor `v[]` é usado para armazenar as velocidades dos N blocos. Supomos aqui que sempre temos

$$\mathtt{x}[0] < \mathtt{x}[1] < \cdots < \mathtt{x}[N-1]. \tag{5}$$

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

#define NMAX 100
#define L 0.1

int read_data(double x[], double v[]);
double min_entry(double t[], int N, int *i);
double time_to_col(double x, double xx, double v, double vv);
double time_to_next_col(double x[], double v[], int N, int *i);

int main() {
  int N, i;
  double x[NMAX], v[NMAX];
  double t;

  N = read_data(x, v);
  t = time_to_next_col(x, v, N, &i);

  printf("Time to next collision: %f\n", t);
  printf("Next collision will be between blocks %d and %d.\n", i - 1, i);

  return 0;
}

int read_data(double x[], double v[]) {
  int i, N;

  scanf("%d", &N);
  for (i = 0; i < N; i++)
    scanf("%lf", &x[i]);

  for (i = 0; i < N; i++)
    scanf("%lf", &v[i]);

  return N;
}
```

```
double min_entry(double t[], int N, int *i_min) {
  int i;
  double min = HUGE_VAL;

  for (i = 1; i < N; i++) {
    if (t[i] < min) {
      min = t[i];
      *i_min = i;
    }
  }

  return min;
}
```

(a) Escreva uma função de protótipo

`double time_to_col(double x, double xx, double v, double vv);`

que recebe a posição `x` e a velocidade `v` de um bloco e a posição `xx` e a velocidade `vv` de um outro bloco, e que devolve o intervalo de tempo até a colisão entre esses dois blocos. Lembre que os blocos têm dimensão $L \times L \times L = 0.1 \times 0.1 \times 0.1$. Caso os blocos não vão colidir, sua função deve devolver o valor `HUGE_VAL`. Por simplicidade, você pode supor neste item que $|\,\texttt{x} - \texttt{xx}\,| > L = 0.1$.

(b) Escreva uma função de protótipo

```
double time_to_next_col(double x[], double v[], int N, int *i);
```

que recebe em `x` as posições dos centros de `N` blocos e em `v` as velocidades desses `N` blocos, e que devolve o intervalo de tempo até a próxima colisão que ocorrerá nesse sistema. Você deve usar a função `min_entry()` dada no enunciado e a função `time_to_col()` do item anterior, mesmo que você não a tenha escrito.