|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA  **KHOA CÔNG NGHỆ THÔNG TIN** |

**ĐỀ THI VÀ BÀI LÀM**

Tên học phần: Toán ứng dụng CNTT

Mã học phần: Hình thức thi: *Tự luận có giám sát*

Đề số: **Đ0003** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*

**Họ tên:** Đoàn Kiều Ngân **Lớp**: 23T\_DT1 **MSSV**: 102230201

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV\_HọTên.pdf và nộp bài thông qua MS Teams.

***Câu 1*** (*2.0 điểm*): Viết chương trình (có sử dụng hàm) thực hiện công việc sau, biết rằng n=100:

, .

a) *(1.0 điểm)* Kiểm tra F(n) có phải số nguyên tố hay không?

|  |
| --- |
| **# Trả lời: Dán code bên dưới:**  #include <stdio.h>  #include <math.h>  // Hàm kiểm tra số nguyên tố  int laSoNguyenTo(unsigned long long n) {      if (n < 2) return 0;      if (n == 2 || n == 3) return 1;      if (n % 2 == 0 || n % 3 == 0) return 0;      for (unsigned long long i = 5; i \* i <= n; i += 6) {          if (n % i == 0 || n % (i + 2) == 0) return 0;      }      return 1;  }  // Hàm tính số Fibonacci thứ k  unsigned long long fibonacci(int k) {      if (k <= 2) return 1;      unsigned long long f1 = 1, f2 = 1, fn;      for (int i = 3; i <= k; i++) {          fn = f1 + f2;          f1 = f2;          f2 = fn;      }      return fn;  }  // Hàm liệt kê và đếm số nguyên tố trong n số Fib đầu tiên  void lietKeVaDemSNTFib(int n) {      unsigned long long f1 = 1, f2 = 1, fn;      int dem = 0;        printf("Cac so Fibonacci la so nguyen to trong %d so dau tien:\n", n);        // Hai số đầu tiên là 1, 1 (không phải số nguyên tố)      for (int i = 3; i <= n; i++) {          fn = f1 + f2;          if (laSoNguyenTo(fn)) {              printf("%llu ", fn);              dem++;          }          f1 = f2;          f2 = fn;            // Chặn lỗi tràn số khi n quá lớn (tùy chọn)          if (fn < f1) break;      }      printf("\nTong cong co %d so nguyen to.\n", dem);  }  int main() {      int n = 100;      unsigned long long fn = fibonacci(n);      printf("F(%d) = %llu\n", n, fn);      if (laSoNguyenTo(fn)) {          printf("F(%d) la so nguyen to.\n", n);      } else {          printf("F(%d) khong phai la so nguyen to.\n", n);      }      printf("\n------------------------------\n");      lietKeVaDemSNTFib(n);      return 0;  }  **# Trả lời: Dán kết quả thực thi vào bên dưới:** |

1. *(1.0 điểm)* Trong n số Fibonacci đầu tiên, hãy liệt kê các số nào là số nguyên tố và đếm tổng số đó.

|  |
| --- |
| **# Trả lời: Dán code vào bên dưới:**  #include <stdio.h>  #include <math.h>  // Hàm kiểm tra số nguyên tố  int laSoNguyenTo(unsigned long long n) {      if (n < 2) return 0;      if (n == 2 || n == 3) return 1;      if (n % 2 == 0 || n % 3 == 0) return 0;      for (unsigned long long i = 5; i \* i <= n; i += 6) {          if (n % i == 0 || n % (i + 2) == 0) return 0;      }      return 1;  }  // Hàm tính số Fibonacci thứ k  unsigned long long fibonacci(int k) {      if (k <= 2) return 1;      unsigned long long f1 = 1, f2 = 1, fn;      for (int i = 3; i <= k; i++) {          fn = f1 + f2;          f1 = f2;          f2 = fn;      }      return fn;  }  // Hàm liệt kê và đếm số nguyên tố trong n số Fib đầu tiên  void lietKeVaDemSNTFib(int n) {      unsigned long long f1 = 1, f2 = 1, fn;      int dem = 0;        printf("Cac so Fibonacci la so nguyen to trong %d so dau tien:\n", n);      for (int i = 3; i <= n; i++) {          fn = f1 + f2;          if (laSoNguyenTo(fn)) {              printf("%llu ", fn);              dem++;          }          f1 = f2;          f2 = fn;            if (fn < f1) break;      }      printf("\nTong cong co %d so nguyen to.\n", dem);  }  int main() {      int n = 100;      unsigned long long fn = fibonacci(n);      printf("F(%d) = %llu\n", n, fn);      if (laSoNguyenTo(fn)) {          printf("F(%d) la so nguyen to.\n", n);      } else {          printf("F(%d) khong phai la so nguyen to.\n", n);      }      printf("\n------------------------------\n");      lietKeVaDemSNTFib(n);      return 0;  }  **# Trả lời: Dán kết quả thực thi vào bên dưới:** |

***Câu 2*** (*1.5 điểm*): Cho ma trận A. Viết chương trình (có sử dụng hàm) thực hiện phân rã ma trận A bằng phương pháp SVD.

|  |
| --- |
| **# Trả lời:** **Dán code vào bên dưới**  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #define MAX 10  #define EPSILON 1e-5  #define MAX\_ITER 1000  // Hàm nhân ma trận A (m x n) với B (n x p)  void nhanMaTran(double A[][MAX], double B[][MAX], double C[][MAX], int m, int n, int p) {      for (int i = 0; i < m; i++) {          for (int j = 0; j < p; j++) {              C[i][j] = 0;              for (int k = 0; k < n; k++) {                  C[i][j] += A[i][k] \* B[k][j];              }          }      }  }  // Hàm chuyển vị ma trận  void chuyenVi(double A[][MAX], double AT[][MAX], int m, int n) {      for (int i = 0; i < m; i++) {          for (int j = 0; j < n; j++) {              AT[j][i] = A[i][j];          }      }  }  // Hàm tính độ dài vector  double doDaiVector(double v[], int n) {      double sum = 0;      for (int i = 0; i < n; i++) {          sum += v[i] \* v[i];      }      return sqrt(sum);  }  // Hàm chuẩn hóa vector  void chuanHoaVector(double v[], int n) {      double len = doDaiVector(v, n);      if (len > EPSILON) {          for (int i = 0; i < n; i++) {              v[i] /= len;          }      }  }  // Hàm tính singular value lớn nhất bằng Power Iteration  double powerIteration(double ATA[][MAX], int n, double v[]) {      double lambda = 0;      double v\_old[MAX];      // Khởi tạo vector ngẫu nhiên      for (int i = 0; i < n; i++) {          v[i] = 1.0;      }      chuanHoaVector(v, n);      for (int iter = 0; iter < MAX\_ITER; iter++) {          // Lưu vector cũ          for (int i = 0; i < n; i++) {              v\_old[i] = v[i];          }          // v = A^T \* A \* v          double temp[MAX] = {0};          for (int i = 0; i < n; i++) {              temp[i] = 0;              for (int j = 0; j < n; j++) {                  temp[i] += ATA[i][j] \* v[j];              }          }          // Tính eigenvalue          lambda = 0;          for (int i = 0; i < n; i++) {              lambda += temp[i] \* v[i];          }          // Chuẩn hóa          for (int i = 0; i < n; i++) {              v[i] = temp[i];          }          chuanHoaVector(v, n);          // Kiểm tra hội tụ          double diff = 0;          for (int i = 0; i < n; i++) {              diff += fabs(v[i] - v\_old[i]);          }          if (diff < EPSILON) {              break;          }      }      return sqrt(fabs(lambda));  }  // Hàm phân rã SVD đơn giản (chỉ tính singular values chính)  void svdDecomposition(double A[][MAX], int m, int n, double S[], double U[][MAX], double VT[][MAX]) {      double AT[MAX][MAX], ATA[MAX][MAX], AAT[MAX][MAX];      printf("\n=== BAT DAU PHAN RA SVD ===\n");      // Bước 1: Tính A^T      printf("\nBuoc 1: Tinh chuyen vi A^T\n");      chuyenVi(A, AT, m, n);      // Bước 2: Tính A^T \* A      printf("Buoc 2: Tinh A^T \* A\n");      nhanMaTran(AT, A, ATA, n, m, n);      // Bước 3: Tìm singular values (eigenvalues của A^T \* A)      printf("Buoc 3: Tim singular values bang Power Iteration\n");      int rank = (m < n) ? m : n;      for (int i = 0; i < rank; i++) {          double v[MAX];          S[i] = powerIteration(ATA, n, v);          // Lưu vector riêng vào V^T          for (int j = 0; j < n; j++) {              VT[i][j] = v[j];          }          // Deflation: trừ đi thành phần đã tìm được          for (int j = 0; j < n; j++) {              for (int k = 0; k < n; k++) {                  ATA[j][k] -= S[i] \* S[i] \* v[j] \* v[k];              }          }            if (S[i] < EPSILON) {              S[i] = 0;              rank = i;              break;          }      }        // Bước 4: Tính U = A \* V \* S^-1      printf("Buoc 4: Tinh ma tran U\n");      for (int i = 0; i < m; i++) {          for (int j = 0; j < rank; j++) {              U[i][j] = 0;              for (int k = 0; k < n; k++) {                  U[i][j] += A[i][k] \* VT[j][k];              }              if (S[j] > EPSILON) {                  U[i][j] /= S[j];              }          }      }        printf("\n=== HOAN THANH PHAN RA SVD ===\n");  }  // Hàm in ma trận  void inMaTran(double A[][MAX], int m, int n, const char\* ten) {      printf("\n%s (%dx%d):\n", ten, m, n);      for (int i = 0; i < m; i++) {          for (int j = 0; j < n; j++) {              printf("%8.4f ", A[i][j]);          }          printf("\n");      }  }  // Hàm in singular values  void inSingularValues(double S[], int n, const char\* ten) {      printf("\n%s:\n", ten);      for (int i = 0; i < n; i++) {          if (S[i] > EPSILON) {              printf("S[%d] = %.4f\n", i, S[i]);          }      }  }  int main() {      int m, n;      double A[MAX][MAX], U[MAX][MAX], S[MAX], VT[MAX][MAX];        printf("========================================\n");      printf("   PHAN RA SVD (Singular Value Decomposition)\n");      printf("========================================\n");        // Input      printf("\nNhap kich thuoc ma tran A (m x n):\n");      printf("Nhap m (so hang): ");      scanf("%d", &m);      printf("Nhap n (so cot): ");      scanf("%d", &n);        if (m <= 0 || n <= 0 || m > MAX || n > MAX) {          printf("Kich thuoc khong hop le!\n");          return 1;      }        printf("\nNhap cac phan tu cua ma tran A:\n");      for (int i = 0; i < m; i++) {          for (int j = 0; j < n; j++) {              printf("A[%d][%d] = ", i, j);              scanf("%lf", &A[i][j]);          }      }        // Khởi tạo ma trận      for (int i = 0; i < MAX; i++) {          S[i] = 0;          for (int j = 0; j < MAX; j++) {              U[i][j] = 0;              VT[i][j] = 0;          }      }        // In ma trận input      inMaTran(A, m, n, "Ma tran A (Input)");        // Thực hiện phân rã SVD      svdDecomposition(A, m, n, S, U, VT);        // Output kết quả      printf("\n========================================\n");      printf("   KET QUA PHAN RA SVD\n");      printf("========================================\n");      printf("\nA = U \* S \* V^T\n");        int rank = (m < n) ? m : n;      inMaTran(U, m, rank, "Ma tran U");      inSingularValues(S, rank, "Singular Values (S)");      inMaTran(VT, rank, n, "Ma tran V^T");        printf("\n========================================\n");        return 0;  }  **# Trả lời:** **Dán kết quả thực thi** vào bên dưới biết rằng , sai số . |

***Câu 3*** (*3.0 điểm*): Cho 18 điểm trong không gian Oxy như sau: *(1,0), (4,0), (6,2), (5,5), (3,6), (0,3), (2,0), (0,2), (3,3), (4,1), (2,2),(4,4), (2,4), (1,2),(1,5),(3,1),(3,5),(5,3).*

1. *(0.5 điểm) Mô tả thuật toán xác định bao lồi của tập điểm trên (dạng sơ đồ khối hoặc mã giả).*

|  |
| --- |
| **# Trả lời:** **dán sơ đồ khối hoặc mã giả:**  Mô tả: Từ tập các điểm theo yêu cầu đề bài ta tìm bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm đề bài  Input: Toạ độ các điểm theo đề bài  Thuật toán:  Bước 1: Sắp xếp các điểm theo toạ độ x tăng dần (nếu cùng toạ độ x thì ta xếp theo toạ độ y tăng dần)  Xác định bao trên Lupper (bước 2, 3)  Bước 2: Bổ sung P1 và P2 vào bao trên Lupper với P1 là điểm bắt đầu  Bước 3: Bổ sung điểm thứ 3 (nếu 3 điểm cuối cùng không tạo thành rẻ phải thì xoá điểm giữa)  Xác định bao dưới Llower (bước 4, 5, 6)  Bước 4: Bổ sung Pn và Pn-1 vào bao dưới Llower với Pn là điểm bắt đầu  Bước 5: Bổ sung điểm thứ 3, nếu 3 điểm không tạo thành rẻ phải thì điểm giữa  Bước 6: Xoá điểm đầu và điểm cuối trong bao dưới Llower  Bước 7: Tập hợp điểm thu được theo chiều kim đồng hồ: Lupper U Llower  Output: Bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm) |

1. *(1.0 điểm)* Viết chương trình tìm bao lồi, sau đó tính cạnh nhỏ nhất của đa giác lồi tìm được.

|  |
| --- |
| **# Trả lời:** **Dán code bên dưới:**  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <float.h>    typedef struct {      double x, y;      char ten[5];  } Diem;    int soSanh(const void \*a, const void \*b) {      Diem \*d1 = (Diem\*)a, \*d2 = (Diem\*)b;      if (d1->x != d2->x)          return (d1->x < d2->x) ? -1 : 1;      return (d1->y < d2->y) ? -1 : (d1->y > d2->y);  }    double tichCoHuong(Diem p1, Diem p2, Diem p3) {      return (p2.x - p1.x) \* (p3.y - p1.y) -             (p2.y - p1.y) \* (p3.x - p1.x);  }    void timBaoLoi(Diem \*tapDiem, int n, Diem \*bao, int \*soDinh) {      qsort(tapDiem, n, sizeof(Diem), soSanh);        Diem \*H = (Diem\*)malloc(2\*n \* sizeof(Diem));      int k = 0;        for (int i = 0; i < n; i++) {          while (k >= 2 && tichCoHuong(H[k-2], H[k-1], tapDiem[i]) <= 0)              k--;          H[k++] = tapDiem[i];      }        for (int i = n-2, t = k+1; i >= 0; i--) {          while (k >= t && tichCoHuong(H[k-2], H[k-1], tapDiem[i]) <= 0)              k--;          H[k++] = tapDiem[i];      }        \*soDinh = k - 1;      for (int i = 0; i < \*soDinh; i++) {          bao[i] = H[i];      }      free(H);  }    double dienTichBaoLoi(Diem \*bao, int soDinh) {      double S = 0.0;      for (int i = 0; i < soDinh; i++) {          int j = (i + 1) % soDinh;          S += bao[i].x \* bao[j].y - bao[j].x \* bao[i].y;      }      return fabs(S) / 2.0;  }    double khoangCach(Diem a, Diem b) {      return sqrt((a.x - b.x)\*(a.x - b.x) + (a.y - b.y)\*(a.y - b.y));  }    void timCapGanNhatTrenBaoLoi(Diem \*bao, int soDinh, Diem \*p1, Diem \*p2, double \*minDist) {      \*minDist = 1e9;      for (int i = 0; i < soDinh; i++) {          for (int j = i+1; j < soDinh; j++) {              double d = khoangCach(bao[i], bao[j]);              if (d < \*minDist) {                  \*minDist = d;                  \*p1 = bao[i];                  \*p2 = bao[j];              }          }      }  }    int main() {      Diem tapDiem[] = {          {1,0, "p1"},          {4,0, "p2"},          {6,2, "p3"},          {5,5, "p4"},          {3,6, "p5"},          {0,3, "p6"},          {2,0, "p7"},          {0,2, "p8"},          {3,3, "p9"},          {4,1, "p10"},          {2,2, "p11"},          {4,4, "p12"},          {2,4, "p13"},          {1,2, "p14"},          {1,5, "p15"},          {3,1, "p16"},          {3,5, "p17"},          {5,3, "p18"},        };      int n = sizeof(tapDiem)/sizeof(tapDiem[0]);        Diem \*bao = (Diem\*)malloc(2 \* n \* sizeof(Diem));      int soDinh;        timBaoLoi(tapDiem, n, bao, &soDinh);        printf("Cac dinh tren bao loi la:\n");      for (int i = 0; i < soDinh; i++) {          printf("%s (%.1f, %.1f)\n", bao[i].ten, bao[i].x, bao[i].y);      }        printf("Dien tich bao loi = %.2f\n", dienTichBaoLoi(bao, soDinh));        Diem p1, p2;      double minDist;      timCapGanNhatTrenBaoLoi(bao, soDinh, &p1, &p2, &minDist);        printf("Cap diem gan nhat tren bao loi: %s (%.1f, %.1f) va %s (%.1f, %.1f)\n",             p1.ten, p1.x, p1.y, p2.ten, p2.x, p2.y);      printf("Khoang cach = %.3f\n", minDist);        free(bao);        return 0;  }  **# Trả lời:** **Dán kết quả thực thi vào bên dưới:** |

c)  *(1.5 điểm)* Xác định diện tích của đa giác bao lồi vừa tìm được. Xác định số lượng các điểm nằm bên trong bao lồi và liệt kê chúng.

|  |
| --- |
| **# Trả lời:** **Dán code bên dưới:**  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <float.h>    typedef struct {      double x, y;      char ten[5];  } Diem;    int soSanh(const void \*a, const void \*b) {      Diem \*d1 = (Diem\*)a, \*d2 = (Diem\*)b;      if (d1->x != d2->x)          return (d1->x < d2->x) ? -1 : 1;      return (d1->y < d2->y) ? -1 : (d1->y > d2->y);  }    double tichCoHuong(Diem p1, Diem p2, Diem p3) {      return (p2.x - p1.x) \* (p3.y - p1.y) -             (p2.y - p1.y) \* (p3.x - p1.x);  }    void timBaoLoi(Diem \*tapDiem, int n, Diem \*bao, int \*soDinh) {      qsort(tapDiem, n, sizeof(Diem), soSanh);        Diem \*H = (Diem\*)malloc(2\*n \* sizeof(Diem));      int k = 0;        for (int i = 0; i < n; i++) {          while (k >= 2 && tichCoHuong(H[k-2], H[k-1], tapDiem[i]) <= 0)              k--;          H[k++] = tapDiem[i];      }        for (int i = n-2, t = k+1; i >= 0; i--) {          while (k >= t && tichCoHuong(H[k-2], H[k-1], tapDiem[i]) <= 0)              k--;          H[k++] = tapDiem[i];      }        \*soDinh = k - 1;      for (int i = 0; i < \*soDinh; i++) {          bao[i] = H[i];      }      free(H);  }    double dienTichBaoLoi(Diem \*bao, int soDinh) {      double S = 0.0;      for (int i = 0; i < soDinh; i++) {          int j = (i + 1) % soDinh;          S += bao[i].x \* bao[j].y - bao[j].x \* bao[i].y;      }      return fabs(S) / 2.0;  }    double khoangCach(Diem a, Diem b) {      return sqrt((a.x - b.x)\*(a.x - b.x) + (a.y - b.y)\*(a.y - b.y));  }    void timCapGanNhatTrenBaoLoi(Diem \*bao, int soDinh, Diem \*p1, Diem \*p2, double \*minDist) {      \*minDist = 1e9;      for (int i = 0; i < soDinh; i++) {          for (int j = i+1; j < soDinh; j++) {              double d = khoangCach(bao[i], bao[j]);              if (d < \*minDist) {                  \*minDist = d;                  \*p1 = bao[i];                  \*p2 = bao[j];              }          }      }  }    int main() {      Diem tapDiem[] = {          {1,0, "p1"},          {4,0, "p2"},          {6,2, "p3"},          {5,5, "p4"},          {3,6, "p5"},          {0,3, "p6"},          {2,0, "p7"},          {0,2, "p8"},          {3,3, "p9"},          {4,1, "p10"},          {2,2, "p11"},          {4,4, "p12"},          {2,4, "p13"},          {1,2, "p14"},          {1,5, "p15"},          {3,1, "p16"},          {3,5, "p17"},          {5,3, "p18"},        };      int n = sizeof(tapDiem)/sizeof(tapDiem[0]);        Diem \*bao = (Diem\*)malloc(2 \* n \* sizeof(Diem));      int soDinh;        timBaoLoi(tapDiem, n, bao, &soDinh);        printf("Cac dinh tren bao loi la:\n");      for (int i = 0; i < soDinh; i++) {          printf("%s (%.1f, %.1f)\n", bao[i].ten, bao[i].x, bao[i].y);      }        printf("Dien tich bao loi = %.2f\n", dienTichBaoLoi(bao, soDinh));        Diem p1, p2;      double minDist;      timCapGanNhatTrenBaoLoi(bao, soDinh, &p1, &p2, &minDist);        printf("Cap diem gan nhat tren bao loi: %s (%.1f, %.1f) va %s (%.1f, %.1f)\n",             p1.ten, p1.x, p1.y, p2.ten, p2.x, p2.y);      printf("Khoang cach = %.3f\n", minDist);        free(bao);        return 0;  }  **# Trả lời:** **Dán kết quả thực thi vào bên dưới:** |

***Câu 4*** (*2.0 điểm*): Cho hàm số

1. *(0.5 điểm) Trình bày thuật toán tối ưu hàm số đã cho* sử dụng phương pháp *gradient descent với momentum*, biết rằng tham số học (learning rate) , hệ số động lượng là .

|  |
| --- |
| **# Trả lời: dán sơ đồ khối hoặc mã giả:**  hàm func(double x) {  trả về hàm f(x)  }  hàm grad(double x) {  trả về đạo hàm f(x)  }  Hàm GB\_momentum(double x) {  Gán double deltax <- 0  Gán double deltaX\_new <- 0  gasn double x\_new <- 0  For (i <- 1, i <- 10000, i++) {  deltaX\_new <- -GAMMA \* grad(x) + alpha \* deltaX  x\_new <- x - GAMMA \* grad(x) + alpha \* deltaX\_new;  deltaX <- deltaX\_new;  Nếu (trị tuyệt đối (x - x\_new) lớn hơn sai số yêu cầu) {  break;  } còn không {  x <- x\_new;  }  }  In ra "Min x = " + x;  } |

1. *(1.5 điểm)* Viết chương trình (có dùng hàm) tính giá trị bé nhất của f(x,y) sử dụng phương pháp *gradient descent với momentum* với số bước lặp *N* và sai số .

|  |
| --- |
| **# Trả lời**: **Dán code vào bên dưới:**  **# Trả lời**: **Dán kết quả thực thi** với điểm khởi , tham số học học (*learning rate*) , hệ số động lượng (*momentum coefficient*) là , số bước lặp và sai số : |

***Câu 5*** (*1.5 điểm*): Một hệ thống có chế độ làm việc ở mỗi giai đoạn vận hành chỉ với các trạng thái từ 1 đến 5. Chế độ làm việc của hệ thống này được mô tả bằng ma trận chuyển như sau:

a) (*0.5 điểm*) Vẽ đồ thị biễu diễn chuỗi Markov tương ứng đã cho

|  |
| --- |
| **# Trả lời:** **Dán kết quả vào bên dưới** |

b) (*1.0 điểm*) Giả sử rằng hệ thống bắt đầu học ở trạng thái 2. Tính xác xuất hệ thống làm việc ở trạng thái 4 *sau 3 bước thời gian vận hành,* vàtrạng thái 2 *sau 5 bước thời gian vận hành*

|  |
| --- |
| **# Trả lời**: **Dán kết quả tính toán vào bên dưới:** |

Đà Nẵng, ngày 4 tháng 12 năm 2025

|  |  |
| --- | --- |
| **GIẢNG VIÊN BIÊN SOẠN ĐỀ THI** | **KHOA CÔNG NGHỆ THÔNG TIN**  **( đã duyệt)** |
|  |  |