```c
// Q2.b.

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "nr.h"
#include "nrutil.h"
#include "math.h"

int main() {
    int N_time = 10000;
    int N_space = 1001;
    int i = 0, j = 0;

    float x0 = 0.0, x1 = 100.0;
    float *a, *b, *c, *r, *u;

    double dx = (x1 - x0) / (N_space - 1.);     // Step in x.
    double dt = 0.025 * dx * dx;                  // Time step.
    double alpha = dt / (dx * dx);

    double *integral = malloc ((N_time + 1) * sizeof(double));
    double *u0 = malloc ((N_time + 1) * sizeof(double));

    for (j = 0; j < N_time + 1; j++) {
        integral[j] = 0;
        u0[j] = 0;
    }

    a = vector(0, N_space + 1);
    b = vector(0, N_space + 1);
    c = vector(0, N_space + 1);
    r = vector(0, N_space + 1);
    u = vector(0, N_space + 1);

    // The tridiagonal matrix.
    for (i = 1; i < N_space + 1; i++) {
        if (i == N_space) a[i] = 0;     // Boundary condition.
        else    a[i] = -alpha;

        if (i == 1) c[i] = -1/(1 + dx);
        else    c[i] = -alpha;

        if (i == 1 || i == N_space) b[i] = 1.0;
        else    b[i] = 1 + 2 * alpha;
    }

    // Initial condition.
    for (i = 1; i < N_space + 1; i++) {
        if (i == 1) u[i] = 0.0;
        else u[i] = 1.0;
    }

    for (j = 1; j < N_time + 1; j++) {
        for (i = 1; i < N_space + 1; i++) {
            if (i == 1) r[i] = 0;
            else if (i == N_space) r[i] = 1;
            else    r[i] = u[i];
        }
```

```
        tridag(a, b, c, r, u, N_space);

        if (j == 7000) {
            for (i = 0; i < N_space; i++) {
                printf("%d\t%d\t%f\t%f\n", j, i, x0 + i * dx, u[i]);
            }
        }

        // "Energy conservation" check. Integration by Simpson's Rule.
        integral[j] = u[1] + u[N_space];
        for (i = 2; i < N_space - 2; i += 2)
            integral[j] = integral[j] + 4 * u[i] + 2 * u[i + 1];
        integral[j] += 4 * u[N_space - 1];
        integral[j] *= dx / 3.;

        u0[j] = u[1];
        //printf("%d\t%f\n", j, u0[j]);
    }

    for (j = 1; j < N_time; j++) {
        if (j % 7000 == 0) printf("%f\t%f\n", j * dt, (( (integral[j+1]-
            integral[j-1])/(2*dt) ) + u0[j])/((integral[j+1]-integral[j-1])/(2
            *dt) ));
    }

    free_vector(a, 0, N_space + 1);
    free_vector(b, 0, N_space + 1);
    free_vector(c, 0, N_space + 1);
    free_vector(r, 0, N_space + 1);
    free_vector(u, 0, N_space + 1);

    free(integral);
    free(u0);

    return 0;
}
```