פתרון מבחן מועד X מחשוב מקבילי ומבוזר סמסטר 2022 א

שאלה מס' 1.1

תיאור של מצב בעייתי	שורה
חסרות פונקציות MPI_Init ו- MPI_Finalize	
הפרמטר הראשון והשני גורמים לגלישה בזיכרון	18
אי התאמה של תגים	22 ,18
הפרמטר הראשון מצביע למקום ללא הקצאה	22
התהליך השלישי לא מקבל את הנתונים	
רק תהליך 1 מדפיס את המערך	24

שאלה מס' 1.2

תיאור של מצב בעייתי	שורה
הדפסה של משתנה max, שלא קשור למשתנים בשורות 10 ו-12	15
המשתנה x הוגדר כפרטי, אין קשר למשתנה x בשורה 5	12
המשתנה tid הוגדר כמשותף, ויכול להידרס על ידי כתיבה של תהליכונים שונים	11

שאלה מס' 1.3

תיאור של מצב בעייתי	שורה
הקצאת מקום לא מספיקה	12
כיוון העתקה לא נכון	13
לא מספיק תהליכונים בשביל לעבור על כול המערך	5 ,16 ,15
הפונקציה **changeString** מקבלת מצביע לזיכרון ב-CPU	18
חישוב לא נכון של הזיכרון להעתקה	20
שחרור זיכרון שלא הוקצה עם cudaMalloc	23

שאלה מס' 2.1 (20 נקודות)

```
// This program will run with 8 processes, a total number of a given cores
// Each process will calculate a half elements of its range
// The process 0 will copy the correspondent values to the second halves of each range
if (rank == 0) {
        setInitialValues(A);
        showArray(A);
}
broadcast(A, P0);
size = N / 16;  // Half of range
if (rank == 0) {
        result = calculateHalfRange(A, size);
        updateHalfRange(A, result, 0)
        updateSecondHalf(A, result, 0);
        for (k = 1;   k  <  numOfProcesses;   k++) {
                recv(result, Pk);
                updateHalfRange(A, result, k)
                updateTheSecondHalf(A, result, k);
        }
        showArray(A);
} else {
        result = calculateHalfRange(A, size);
        send(result, P0);
   }
```

שאלה מס' 2.2 (50 נקודות)

```
#define N 32
#define NPROC 8

int f(int x) {
	return 2*x;
}

int main(int argc, char* argv[]){
	int A[N];
	int  rank;
	int  numProc;
	MPI_Status status ;
	int size, i, j, start;


	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &numProc);

	if (rank == 0 && (numProc != NPROC || N%NPROC != 0)) {
		printf("This program runs with 8 processes, N/2 is divided by nproc
without remainder\n");
		MPI_Abort(MPI_COMM_WORLD, 0);
	}

	// Assign initial values, just to demonstrate the functionality
	for (i = 0;   i < N;   i++) {
		A[i] = rand()%100;;
	}

	// Display the initial array
	if (rank == 0) {
		for (i = 0;  i < N;   i++)
			printf("%d ", A[i]);
		printf("\n");
	}

	// Send the array to all processes
	MPI_Bcast(A, N, MPI_INT, 0, MPI_COMM_WORLD);

	// Number of members of A to be managed by each process
	size = N / NPROC / 2;

	// Each process updates a half of its range
	start = rank*size;
	for (i = start;   i < start + size;  i++) {
		int value1 = f(A[i]);
		int value2 = f(A[N-1-i]);
```

```
            A[i] = value1 > value2 ? value1 : value2;
        }

        // Send the result to the process 0
        if (rank == 0) {
            for (j = 0;   j < size;  j++)
                A[N-1-j] = A[j];
            for (i = 1;  i < numProc;   i++) {
                MPI_Recv(A + i*size, size, MPI_INT, i, 0, MPI_COMM_WORLD, &status);
            // Copy the result to the remaining part the process i is responsible for
                for (j = 0;   j < size;  j++)
                    A[N-1-(i*size + j)] = A[i*size + j];
            }
        } else
            MPI_Send(A + start, size, MPI_INT, 0, 0, MPI_COMM_WORLD);
        // Display an array with a new values
        if (rank == 0)
            for (i = 0;  i < N;   i++)
                printf("%d ", A[i]);

        MPI_Finalize();
        return 0;
}
```