Assignment 3

Name: Diptangshu Dey

Roll No. 20CS8018

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int *genarray(int n) {
  int *A, i, S;

  A = (int *)malloc((n + 1) * sizeof(int));
  S = 0;
  printf("The input array:\n");
  for (i = 1; i <= n; ++i) {
    A[i] = 1 + rand() % 99;
    printf("%-2d  ", A[i]);
    if (i % 20 == 0)
      printf("\n");
    S += A[i];
  }
  if (i % 20 != 1)
    printf("\n");
  // A[0] = 1 + rand() % S;
  // A[0] = (S/2) + (rand() % (S/2));
  A[0] = (2 * S / 3) + (rand() % (S / 3));
  if ((A[0] & 1) != (S & 1))
    --A[0];
  if (rand() & 1)
    A[0] = -A[0];
  printf("S = %d\n", S);
  printf("T = %d\n", A[0]);
  return A;
}

void realizable(int A[], int n, int T) {
  int S, s, i, j;
  char **P; /* P[i][j] stores 0 if unrealizable, or +/- if realizable */

  S = 0;
  for (i = 1; i <= n; ++i)
    S += A[i];
  P = (char **)malloc((n + 1) * sizeof(char *));
  for (i = 0; i <= n; ++i) {
    P[i] = (char *)malloc((2 * S + 1) * sizeof(char));
    for (j = -S; j <= S; ++j)
      P[i][j + S] = '\0';
```

```
  }
  P[0][S] = '.';
  s = 0;
  for (i = 1; i <= n; ++i) {
    s += A[i];
    for (j = -s; j <= s; ++j) {
      if ((j - A[i] >= -S) && (P[i - 1][j - A[i] + S]))
        P[i][j + S] = '+';
      else if ((j + A[i] <= S) && (P[i - 1][j + A[i] + S]))
        P[i][j + S] = '-';
    }
  }
  if (P[n][T + S])
    printf("   The value %d is realizable\n", T);
  else
    printf("   The value %d is not realizable\n", T);
  for (i = 0; i <= n; ++i)
    free(P[i]);
  free(P);
}

/* sol[] stores a sequence of n + and - signs to indicate the choice of
   the signs of the array elements. */
void printsol(int A[], int n, char *sol) {
  int S, i;

  S = 0;
  for (i = 1; i <= n; ++i) {
    printf("%c%d", sol[i], A[i]);
    if (sol[i] == '+')
      S += A[i];
    else if (sol[i] == '-')
      S -= A[i];
    else
      fprintf(stderr, "*** Error: operator %d is %d\n", i, (int)sol[i]);
  }
  printf(" =  %d\n", S);
}

void showone(int A[], int n, int T) {
  int S, s, i, j;
  char **P, *sol; /* P[i][j] stores 0 if unrealizable, or +/- if realizable */

  S = 0;
  for (i = 1; i <= n; ++i)
    S += A[i];
  P = (char **)malloc((n + 1) * sizeof(char *));
  for (i = 0; i <= n; ++i) {
    P[i] = (char *)malloc((2 * S + 1) * sizeof(char));
    for (j = -S; j <= S; ++j)
      P[i][j + S] = '\0';
  }
  P[0][S] = '.';
  s = 0;
  for (i = 1; i <= n; ++i) {
    s += A[i];
    for (j = -s; j <= s; ++j) {
      if ((j - A[i] >= -S) && (P[i - 1][j - A[i] + S]))
```

```
        P[i][j + S] = '+';
      else if ((j + A[i] <= S) && (P[i - 1][j + A[i] + S]))
        P[i][j + S] = '-';
    }
  }
  if (P[n][T + S]) {
    sol = (char *)malloc((n + 1) * sizeof(char));
    j = T;
    i = n;
    while (i > 0) {
      sol[i] = P[i][j + S];
      if (sol[i] == '+')
        j -= A[i];
      else
        j += A[i];
      --i;
    }
    printf("    Solution: ");
    printsol(A, n, sol);
    free(sol);
  } else
    printf("   The value %d is not realizable\n", T);
  for (i = 0; i <= n; ++i)
    free(P[i]);
  free(P);
}

/* Generate all possible solution vectors recursively */
void gensol(int **P, int **Q, int *A, char **sol, int i, int j, int S) {
  int npos, nneg, k;

  if (i <= 0)
    return;
  npos = P[i][j + S];
  nneg = Q[i][j + S];
  for (k = 0; k < npos; ++k)
    sol[k][i] = '+';
  for (k = 0; k < nneg; ++k)
    sol[k + npos][i] = '-';
  if (npos)
    gensol(P, Q, A, sol, i - 1, j - A[i], S);
  if (nneg)
    gensol(P, Q, A, sol + npos, i - 1, j + A[i], S);
}

void showall(int A[], int n, int T) {
  int S, s, i, j, nsol;
  char **sol;
  int **P; /* P[i][j] stores how many times j is realizable with +a[i] */
  int **Q; /* Q[i][j] stores how many times j is realizable with -a[i] */
           /* Total number of realizations of j is P[i][j] + Q[i][j] */

  S = 0;
  for (i = 1; i <= n; ++i)
    S += A[i];
  P = (int **)malloc((n + 1) * sizeof(int *));
  Q = (int **)malloc((n + 1) * sizeof(int *));
  for (i = 0; i <= n; ++i) {
```

```
    P[i] = (int *)malloc((2 * S + 1) * sizeof(int));
    Q[i] = (int *)malloc((2 * S + 1) * sizeof(int));
    for (j = -S; j <= S; ++j)
      P[i][j + S] = Q[i][j + S] = 0;
  }
  P[0][S] = Q[0][S] = 1; /* Initialization of Row 0 */
  P[1][A[1] + S] = 1;
  Q[1][-A[1] + S] = 1; /* Preferable to initialize Row 1 too */
  s = A[1];
  for (i = 2; i <= n; ++i) {
    s += A[i];
    for (j = -s; j <= s; ++j) {
      if (j - A[i] >= -S)
        P[i][j + S] = P[i - 1][j - A[i] + S] + Q[i - 1][j - A[i] + S];
      if (j + A[i] <= S)
        Q[i][j + S] = P[i - 1][j + A[i] + S] + Q[i - 1][j + A[i] + S];
    }
  }
  nsol = P[n][T + S] + Q[n][T + S];
  printf("    Number of solutions = %d\n", nsol);
  if (nsol) {
    /* Allocate memory for n solution vectors */
    sol = (char **)malloc(nsol * sizeof(char *));
    for (i = 0; i < nsol; ++i)
      sol[i] = (char *)malloc((n + 1) * sizeof(char));

    /* Recursively generate all solution vectors */
    gensol(P, Q, A, sol, n, T, S);

    /* Print all the nsol solutions */
    for (i = 0; i < nsol; ++i) {
      printf("    Sol %3d : ", i + 1);
      printsol(A, n, sol[i]);
      free(sol[i]);
    }
    free(sol);
  }
  for (i = 0; i <= n; ++i) {
    free(P[i]);
    free(Q[i]);
  }
  free(P);
  free(Q);
}

int main(int argc, char *argv[]) {
  int n, *A, T;

  srand((unsigned int)time(NULL));
  if (argc > 1)
    n = atoi(argv[1]);
  else
    scanf("%d", &n);
  printf("n = %d\n", n);

  A = genarray(n);
  T = A[0];
```

```
    printf("\n+++ Part 1: Realizability check\n");
    realizable(A, n, T);

    printf("\n+++ Part 2: One solution\n");
    showone(A, n, T);

    printf("\n+++ Part 3: All solutions\n");
    showall(A, n, T);

    exit(0);
}
```

Output:

```
❯ ./a.out
10
n = 10
The input array:
40  62  16  48  56  34  68  5   48  75
S = 452
T = -436

+++ Part 1: Realizability check
   The value -436 is not realizable

+++ Part 2: One solution
   The value -436 is not realizable

+++ Part 3: All solutions
    Number of solutions = 0
```