PROGRAM NO:5

Compress an Image using Harr Transform.

PROGRAM

```
# include <stdlib.h>
# include <stdio.h>
# include <math.h>
# include "haar.h"

int main ( );
void test01 ( );
void test02 ( );

/******************************************************************************/

int main ( )
{
  timestamp ( );
  printf ( "\n" );
  printf ( "HAAR_PRB\n" );
  printf ( "  C version\n" );
  printf ( "  Test the HAAR library.\n" );

  test01 ( );
  test02 ( );
/*
  Terminate.
*/
  printf ( "\n" );
  printf ( "HAAR_PRB\n" );
  printf ( "  Normal end of execution.\n" );
  printf ( "\n" );
  timestamp ( );

  return 0;
}
/******************************************************************************/

void test01 ( )

/******************************************************************************/

{
  double err;
  int i;
  int n;
  int seed;
  double *u;
  double *v;
  double *w;

  printf ( "\n" );
```

```
  printf ( "TEST01\n" );
  printf ( "  HAAR_1D computes the Haar transform of a vector.\n" );
/*
  Random data.
*/
  n = 16;
  seed = 123456789;
  u = r8vec_uniform_01_new ( n, &seed );
  v = r8vec_copy_new ( n, u );

  haar_1d ( n, v );

  w = r8vec_copy_new ( n, v );
  haar_1d_inverse ( n, w );

  printf ( "\n" );
  printf ( "   i      U(i)        H(U)(i)  Hinv(H(U))(i)\n" );
  printf ( "\n" );
  for ( i = 0; i < n; i++ )
  {
    printf ( "  %2d  %10f  %10f  %10f\n", i, u[i], v[i], w[i] );
  }

  free ( u );
  free ( v );
  free ( w );
/*
  Constant signal.
*/
  n = 8;
  u = r8vec_ones_new ( n );
  v = r8vec_copy_new ( n, u );

  haar_1d ( n, v );

  w = r8vec_copy_new ( n, v );
  haar_1d_inverse ( n, w );

  printf ( "\n" );
  printf ( "   i      U(i)        H(U)(i)  Hinv(H(U))(i)\n" );
  printf ( "\n" );
  for ( i = 0; i < n; i++ )
  {
    printf ( "  %2d  %10f  %10f  %10f\n", i, u[i], v[i], w[i] );
  }
  free ( u );
  free ( v );
  free ( w );
/*
  Linear signal.
*/
  n = 16;
  u = r8vec_linspace_new ( n, 1.0, ( double ) n );
  v = r8vec_copy_new ( n, u );

  haar_1d ( n, v );
```

```
  w = r8vec_copy_new ( n, v );
  haar_1d_inverse ( n, w );

  printf ( "\n" );
  printf ( "   i      U(i)        H(U)(i)  Hinv(H(U))(i)\n" );
  printf ( "\n" );
  for ( i = 0; i < n; i++ )
  {
    printf ( "  %2d  %10f  %10f  %10f\n", i, u[i], v[i], w[i] );
  }

  free ( u );
  free ( v );
  free ( w );
/*
  Quadratic data.
*/
  n = 8;
  u = ( double * ) malloc ( n * sizeof ( double ) );
  u[0] = 25.0;
  u[1] = 16.0;
  u[2] = 9.0;
  u[3] = 4.0;
  u[4] = 1.0;
  u[5] = 0.0;
  u[6] = 1.0;
  u[7] = 4.0;
  v = r8vec_copy_new ( n, u );

  haar_1d ( n, v );

  w = r8vec_copy_new ( n, v );
  haar_1d_inverse ( n, w );

  printf ( "\n" );
  printf ( "   i      U(i)        H(U)(i)  Hinv(H(U))(i)\n" );
  printf ( "\n" );
  for ( i = 0; i < n; i++ )
  {
    printf ( "  %2d  %10f  %10f  %10f\n", i, u[i], v[i], w[i] );
  }

  free ( u );
  free ( v );
  free ( w );
/*
  N not a power of 2.
*/
  n = 99;
  seed = 123456789;
  u = r8vec_uniform_01_new ( n, &seed );

  v = r8vec_copy_new ( n, u );
  haar_1d ( n, v );

  w = r8vec_copy_new ( n, v );
  haar_1d_inverse ( n, w );
```

```
  err = r8vec_diff_norm ( n, u, w );

  printf ( "\n" );
  printf ( "  For N = %d, ||u-haar_1d_inverse(haar_1d(u))|| = %g\n", n, err );

  free ( u );
  free ( v );
  free ( w );

  return;
}
/******************************************************************************/

void test02 ( )

/******************************************************************************/

{
  double err;
  int i;
  int j;
  int m = 16;
  int n = 4;
  int seed;
  double *u;
  double *v;
  double *w;

  printf ( "\n" );
  printf ( "TEST02\n" );
  printf ( "  HAAR_2D computes the Haar transform of an array.\n" );
  printf ( "  HAAR_2D_INVERSE inverts the transform.\n" );
/*
  Demonstrate successful inversion.
*/
  seed = 123456789;
  u = r8mat_uniform_01_new ( m, n, &seed );

  r8mat_print ( m, n, u, "  Input array U:" );

  v = r8mat_copy_new ( m, n, u );

  haar_2d ( m, n, v );

  r8mat_print ( m, n, v, "  Transformed array V:" );

  w = r8mat_copy_new ( m, n, v );

  haar_2d_inverse ( m, n, w );

  r8mat_print ( m, n, w, "  Recovered array W:" );

  free ( u );
  free ( v );
  free ( w );
/*
```

```
  M, N not powers of 2.
*/
  m = 37;
  n = 53;
  seed = 123456789;
  u = r8mat_uniform_01_new ( m, n, &seed );

  v = r8mat_copy_new ( m, n, u );
  haar_2d ( m, n, v );

  w = r8mat_copy_new ( m, n, v );
  haar_2d_inverse ( m, n, w );

  err = r8mat_dif_fro ( m, n, u, w );

  printf ( "\n" );
  printf ( "  M = %d, N = %d, ||haar_2d_inverse(haar_2d(u))-u|| = %g\n", m, n, err );

  free ( u );
  free ( v );
  free ( w );

  return;
}
```

RESULT

Program to perform harr transform is executed succesfully and required output is obtained.

OUTPUT

TEST01
HAAR_1D computes the Haar transform of a vector.

i	U(i)	H(U)(i)	Hinv(H(U))(i)
0	0.218418	1.639767	0.218418
1	0.956318	0.067684	0.956318
2	0.829509	0.607044	0.829509
3	0.561695	-0.270838	0.561695
4	0.415307	-0.108234	0.415307
5	0.066119	0.056946	0.066119
6	0.257578	0.083264	0.257578
7	0.109957	0.178427	0.109957
8	0.043829	-0.521774	0.043829
9	0.633966	0.189373	0.633966
10	0.061727	0.246913	0.061727
11	0.449539	0.104384	0.449539
12	0.401306	-0.417290	0.401306
13	0.754673	-0.274224	0.754673
14	0.797287	-0.249868	0.797287
15	0.001838	0.562467	0.001838

i	U(i)	H(U)(i)	Hinv(H(U))(i)
0	1.000000	2.828427	1.000000
1	1.000000	0.000000	1.000000
2	1.000000	0.000000	1.000000
3	1.000000	0.000000	1.000000
4	1.000000	0.000000	1.000000
5	1.000000	0.000000	1.000000
6	1.000000	0.000000	1.000000
7	1.000000	0.000000	1.000000

i	U(i)	H(U)(i)	Hinv(H(U))(i)
0	1.000000	34.000000	1.000000
1	2.000000	-16.000000	2.000000
2	3.000000	-5.656854	3.000000
3	4.000000	-5.656854	4.000000
4	5.000000	-2.000000	5.000000
5	6.000000	-2.000000	6.000000
6	7.000000	-2.000000	7.000000
7	8.000000	-2.000000	8.000000
8	9.000000	-0.707107	9.000000
9	10.000000	-0.707107	10.000000
10	11.000000	-0.707107	11.000000
11	12.000000	-0.707107	12.000000
12	13.000000	-0.707107	13.000000
13	14.000000	-0.707107	14.000000
14	15.000000	-0.707107	15.000000
15	16.000000	-0.707107	16.000000

i	U(i)	H(U)(i)	Hinv(H(U))(i)
0	25.000000	21.213203	25.000000
1	16.000000	16.970563	16.000000
2	9.000000	14.000000	9.000000
3	4.000000	-2.000000	4.000000
4	1.000000	6.363961	1.000000
5	0.000000	3.535534	0.000000
6	1.000000	0.707107	1.000000
7	4.000000	-2.121320	4.000000

For N = 99, ||u-haar_1d_inverse(haar_1d(u))|| = 3.33568e-15

TEST02

HAAR_2D computes the Haar transform of an array.
HAAR_2D_INVERSE inverts the transform.

Input array U:

Col: / Row	0	1	2	3
0:	0.218418	0.897504	0.861216	0.825003
1:	0.956318	0.350752	0.453794	0.824660
2:	0.829509	0.094545	0.911977	0.061862
3:	0.561695	0.013617	0.597917	0.710781
4:	0.415307	0.859097	0.188955	0.088283
5:	0.066119	0.840847	0.761492	0.777994
6:	0.257578	0.123104	0.396988	0.745303
7:	0.109957	0.007512	0.185314	0.308675
8:	0.043829	0.260303	0.574366	0.899373
9:	0.633966	0.912484	0.367027	0.763537
10:	0.061727	0.113664	0.617205	0.761731
11:	0.449539	0.351629	0.361529	0.406970
12:	0.401306	0.822887	0.212930	0.938749
13:	0.754673	0.267132	0.714471	0.562088
14:	0.797287	0.692066	0.117707	0.017820
15:	0.001838	0.561662	0.299329	0.501103

Transformed array V:

Col: / Row	0	1	2	3
0:	3.818003	-0.386034	-0.107788	-0.277843
1:	0.007521	-0.138549	0.188370	0.283179
2:	0.536878	-0.097469	0.547781	0.197526
3:	-0.014880	-0.505449	-0.015094	-0.059040
4:	0.401441	0.060372	-0.479618	-0.378976
5:	0.465916	0.375694	-0.514575	0.196521
6:	0.332723	0.104288	-0.191260	-0.187931
7:	0.421356	-0.324784	0.184047	-0.166718
8:	0.076586	-0.211748	-0.642326	0.203540
9:	0.004908	0.241689	0.093443	0.481489
10:	-0.316363	0.576181	0.165469	0.058587
11:	0.322269	-0.136150	0.016015	-0.112477
12:	-0.317895	-0.560556	0.031022	0.035751

13:	-0.005423	-0.437067	-0.074924	-0.049542
14:	0.027403	0.115707	-0.454561	-0.439101
15:	0.092259	0.562418	0.332522	0.150830

Recovered array W:

Row \ Col:	0	1	2	3
0:	0.218418	0.897504	0.861216	0.825003
1:	0.956318	0.350752	0.453794	0.824660
2:	0.829509	0.094545	0.911977	0.061862
3:	0.561695	0.013617	0.597917	0.710781
4:	0.415307	0.859097	0.188955	0.088283
5:	0.066119	0.840847	0.761492	0.777994
6:	0.257578	0.123104	0.396988	0.745303
7:	0.109957	0.007512	0.185314	0.308675
8:	0.043829	0.260303	0.574366	0.899373
9:	0.633966	0.912484	0.367027	0.763537
10:	0.061727	0.113664	0.617205	0.761731
11:	0.449539	0.351629	0.361529	0.406970
12:	0.401306	0.822887	0.212930	0.938749
13:	0.754673	0.267132	0.714471	0.562088
14:	0.797287	0.692066	0.117707	0.017820
15:	0.001838	0.561662	0.299329	0.501103

M = 37, N = 53, ||haar_2d_inverse(haar_2d(u))-u|| = 2.54887e-14

HAAR_PRB
Normal end of execution.