**Algorithm of all filters:**

Function loadImage () : #this function takes from the user name of file and loaded it

declare imageFileName[100]

output ("Please, enter file name of the image to process:")

input (imageFileName)

Add to it .bmp extension and load image

End function

Function saveImage () : #this function saves the file after modifying it by filters

declare imageFileName[100]

output ("Enter the target image file name: ")

input ( imageFileName)

Add to it .bmp extension and load image

output ("your image ", imageFileName, "is ready :) .... ")

End function

Function bw\_filter(): #first filter black and white

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

if (image[i][j] > 127):

assign image[i][j] = 255

else:

assign image[i][j] = 0

End for

End for

End function

Function invert\_filter() : #second filter invert filter

assign av, n = 0;

for (i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign n= n + image[i][j]

End for

assign av = (n/(265\*265))

End for

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign x, res, n = 0

if (image[i][j] > av-40)

assign n = image[i][j]

assign res = 255 - n

assign image[i][j] = res

else

assign x = image[i][j] +255

if (x >= 256)

assign image[i][j] = 255

else

assign n = image[i][j]

assign res = 255 + n

assign image[i][j] = res

End for

End for

End function

Function merge\_filter() : #filter 3

declare mimage[256][256]

declare image2[256][256]

declare imageFileName2[100]

Get gray scale image 2 file name

output ("Please, enter file name of the second image to merge:")

input (imageFileName2)

Add to it .bmp extension and load image

for (i = 0 To i < SIZE)

for ( j = 0 To j < SIZE)

assign mimage[i][j] = ((image[i][j])+(image2[i][j]))/2

End two for

for (int i = 0; i < SIZE; i++)

for (int j = 0; j< SIZE; j++)

assign image[i][j] = mimage[i][j]

End for

End for

End function

Function flib\_filter() : #filter 4

declare nimage [256][256]

for ( i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign nimage[255-j][i] = image[j][i]

End for

End for

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign image[i][j] = nimage[i][j]

End for

End for

End function

Function d\_filter() : #filter 5 (a-Darken filter)

assign n,res = 0

for (i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign n = image[i][j]

assign res = n

assign res = res\*0.5

assign image[i][j] = res

End for

End for

End function

Function l\_filter() : # filter 5 (b- lighten filter)

assign x,n,res = 0;

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign x = image[i][j]\*1.5

if (x > 255)

assign image[i][j] = 255

else

assign n = image[i][j]

assign res = n

assign res = res \*1.5

assign image[i][j] = res

End for

End for

End function

Function rotate\_filter() : #filter 6

declare nimage[256][256]

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

assign nimage[i][j] = image[255-j][i]

End for

End for

for ( i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = nimage[i][j]

End for

End for

End function

Funcyion detect\_edges() : #filter 7

assign gx\_m[3][3]={{-1,0,1},{-2,0,2},{-1,0,1}}

assign gy\_m[3][3]={{-1,-2,-1},{0,0,0},{1,2,1}}

declare gx ,gy , pixel\_v=0

declare gimage[256][256]

declare dy[256][256]

for (i = 0 To i<SIZE)

for ( j = 0 To j<SIZE)

assign gx=0

assign gy=0;

for (x = 0 To x<3)

for (y = 0 To y<3)

assign gx = gx + ((image[(i-1)+x][(j-1)+y])\*(gx\_m[x][y]))

assign gy = gy + ((image[(i-1)+x][(j-1)+y])\*(gy\_m[x][y]))

End for

End for

assign pixel\_v = sqrt((pow(gx,2))+(pow(gy,2)))

if( pixel\_v > 255 )

assign gimage[i][j] = 255

else

assign gimage[i][j] = pixel\_v

End for

End for

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE)

if( gimage[i][j] != 255 )

assign gimage[i][j] = 0

End for

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = gimage[i][j]

End for

End for

End function

Function zoom\_filter(s, start2, end1, end2) : #filter 8

declare zimage[256][256]

declare zi, zj

for ( i = s and zi=0 To i < end1 and zi<SIZE i++,zi+=2)

for (j = start2 and zj=0 To j < end2 and zj<SIZE; j++,zj+=2)

assign zimage[zi][zj] = image[i][j]

assign zimage[zi][zj+1] = image[i][j]

assign zimage[zi+1][zj] = image[i][j]

assign zimage[zi+1][zj+1] = image[i][j]

assign j = j +1

assign zj = zj + 2

End for

assign i = i +1

assign zi = zi + 2

End for

for ( i = 0 To i < SIZE)

for ( j = 0 To j< SIZE )

assign image[i][j] = zimage[i][j]

End for

End for

End function

Function shrink2\_filter() : #filter 9 (1/2)

assign simage[128][128]

declare si, sj

for (i = 0 and si=0 To i < 128 and si<SIZE)

for (j = 0 and sj=0 To j < 128 and sj<SIZE)

assign simage[i][j] = ((image[si][sj])+(image[si+1][sj+1]))/2

assign simage[i+1][j] = ((image[si+1][sj])+(image[si+1][sj+1]))/2

assign j = j +1

assign sj = sj + 2

End for

assign i = i +1

assign si = si + 2

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = 255

End for

End for

for (i = 0 To i < 128)

for (j = 0 To j< 128)

assign image[i][j] = simage[i][j]

End for

End for

End function

Function shrink3\_filter() : #filter 9 (1/3)

assign simage[85][85]

declare si, sj

for (i = 0 and si=0 To i < 85 and si<SIZE)

for (j = 0 and sj=0 To j < 85 and sj<SIZE)

assign simage[i][j] = ((image[si][sj])+(image[si+1][sj+1]) +(image[si+2][sj+2])))/3

assign simage[i+1][j] = ((image[si+1][sj])+(image[si+1][sj+1]) +(image[si+2][sj+2])))/3

assign j = j +1

assign sj = sj + 3

End for

assign i = i +1

assign si = si + 3

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = 255

End for

End for

for (i = 0 To i < 85)

for (j = 0 To j< 85)

assign image[i][j] = simage[i][j]

End for

End for

End function

Function shrink4\_filter() : #filter 9 (1/4)

assign simage[85][85] , declare si, sj

for (i = 0 and si=0 To i < 85 and si<SIZE)

for (j = 0 and sj=0 To j < 85 and sj<SIZE)

assign simage[i][j]=(image[si][sj]+image[si+1][sj+1]+image[si+2][sj+2]+image[si+3][sj+3])/4

assign simage[i+1][j](image[si+1][sj] + image[si+1][sj+1] + image[si+2][sj+2] +

image[si+3][sj+3])/4

assign j = j +1

assign sj = sj + 3

End for

assign i = i +1

assign si = si + 3

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = 255

End for

End for

for (i = 0 To i < 85)

for (j = 0 To j< 85)

assign image[i][j] = simage[i][j]

End for

End for

End function

void mright\_filter() : #filter a (1/2 mirror filter right)

declare mi,mj

declare nimage[256][256]

for (i = 0 To i < SIZE)

for (j = 128 To j <SIZE)

assign nimage[i][j] = image[i][j]

End for

End for

for (i = 0 and mi =0 To i < SIZE and mi < SIZE)

for (j = 128 and mj = 128 To j<SIZE and mj>= 0)

assign nimage[mi][mj] = image[i][j]

assign j = j + 1

assign mj = mj – 1

End for

assign i = i + 1

assign mi = mi + 1

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = nimage[i][j]

End for

End for

End function

Function downm\_filter() : #filter a (1/2 mirror filter down)

declare mi,mj

declare nimage[256][256]

for (i = 0 To i < SIZE)

for (j = 128 To j <SIZE)

assign nimage[i][j] = image[i][j]

End for

End for

for (i = 128 and mi = 128 To i < SIZE and mi >= 0)

for (j = 0 and mj = 0 To j < SIZE and mj < SIZE)

assign nimage[mi][mj] = image[i][j]

assign j = j + 1

assign mj = mj + 1

End for

assign i = i + 1

assign mi = mi - 1

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = nimage[i][j]

End for

End for

End function

Function mup\_filter() : #filter a (1/2 mirror filter up)

declare mi,mj

declare nimage[256][256]

for (i = 0 To i < SIZE)

for (j = 128 To j <SIZE)

assign nimage[i][j] = image[i][j]

End for

End for

for (i = 128 and mi = 128 To i >= 0 and mi >= 0)

for (j = 0 and mj = 0 To j < SIZE and mj < SIZE)

assign nimage[mi][mj] = image[i][j]

assign j = j + 1

assign mj = mj + 1

End for

assign i = i - 1

assign mi = mi + 1

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = nimage[i][j]

End for

End for

End function

Function shuffle\_filter() : #filter b

declare simage1[128][128]

declare simage2[128][128]

declare simage3[128][128]

declare simage4[128][128]

declare sch[4]

for (i = 0 and x=0 To i<128 and x < 128)

for (j = 0 and y = 0 To j<128 and y < 128)

assign simage1[x][y] = image[i][j]

assign j = j + 1

assign y = y + 1

End for

assign i = i + 1

assign x = x + 1

End for

for (i = 0 and x=0 To i<128 and x < 128)

for (j = 0 and y = 0 To j<128 and y < 128)

assign simage2[x][y] = image[i][j]

assign j = j + 1

assign y = y + 1

End for

assign i = i + 1

assign x = x + 1

End for

for (i = 0 and x=0 To i<128 and x < 128)

for (j = 0 and y = 0 To j<128 and y < 128)

assign simage3[x][y] = image[i][j]

assign j = j + 1

assign y = y + 1

End for

assign i = i + 1

assign x = x + 1

End for

for (i = 0 and x=0 To i<128 and x < 128)

for (j = 0 and y = 0 To j<128 and y < 128)

assign simage4[x][y] = image[i][j]

assign j = j + 1

assign y = y + 1

End for

assign i = i + 1

assign x = x + 1

End for

declare sch1

output("New order of quarters ? ...\n")

output("please enter the order by one digit in once >>>")

for (int i = 0 ; To i < 4)

input (sch1)

while (sch1 < 1 or sch1 > 4)

output ("your input is not valid , please enter 1 or 2 or 3 or 4 only\n ")

input (sch1)

End while

assign sch[i] =sch1

End for

//first quarter///////////////////////////

for (i = 0 and x= 0 To i<128 and x<128)

for (j = 0 and y = 0 To j<128 and y < 128)

if ( sch[0] == 1 ):

assign image[i][j] = simage1[x][y]

else if ( sch[0] == 2 ):

assign image[i][j] = simage2[x][y]

else if ( sch[0] == 3 ):

assign image[i][j] = simage3[x][y];

else if ( sch[0] == 4 ):

assign image[i][j] = simage4[x][y]

assign j = j + 1

assign y = y + 1

End for

End for

//second quarter///////////////////////////

for (i = 0 and x= 0 To i<128 and x<128)

for (j = 0 and y = 0 To j<128 and y < 128)

if ( sch[1] == 1 ):

assign image[i][j] = simage1[x][y]

else if ( sch[1] == 2 ):

assign image[i][j] = simage2[x][y]

else if ( sch[1] == 3 ):

assign image[i][j] = simage3[x][y];

else if ( sch[1] == 4 ):

assign image[i][j] = simage4[x][y]

assign j = j + 1

assign y = y + 1

End for

End for

//third quarter///////////////////////////

for (i = 0 and x= 0 To i<128 and x<128; i++,x++)

for (j = 0 and y = 0 To j<128 and y < 128 ; j++,y++)

if ( sch[2] == 1 ):

assign image[i][j] = simage1[x][y]

else if ( sch[2] == 2 ):

assign image[i][j] = simage2[x][y]

else if ( sch[2] == 3 ):

assign image[i][j] = simage3[x][y];

else if ( sch[2] == 4 ):

assign image[i][j] = simage4[x][y]

assign j = j + 1

assign y = y + 1

End for

End for

//fourth quarter///////////////////////////

for (i = 0 and x= 0 To i<128 and x<128)

for (j = 0 and y = 0 To j<128 and y < 128)

if ( sch[3] == 1 ):

assign image[i][j] = simage1[x][y]

else if ( sch[3] == 2 ):

assign image[i][j] = simage2[x][y]

else if ( sch[3] == 3 ):

assign image[i][j] = simage3[x][y];

else if ( sch[3] == 4 ):

assign image[i][j] = simage4[x][y]

assign j = j + 1

assign y = y + 1

End for

End for

End function

Function blur\_filter(): # filter c

declare bimage[256][256]

for (i = 0 To i<SIZE)

for (j = 0 To j<SIZE)

bimage[i][j] = ((image[i][j])+(image[i][j+1])+(image[i][j+2])+

(image[i][j+3])+(image[i][j+4])+(image[i+1][j])+(image[i+1][j+1])+(image[i+1][j+2])+(image[i+1][j+3])+(image[i+1][j+4])+(image[i+2][j])+(image[i+2][j+1])+(image[i+2][j+2])+(image[i+2][j+3])+(image[i+2][j+4])+(image[i+3][j])+(image[i+3][j+1])+(image[i+3][j+2])+(image[i+3][j+3])+(image[i+3][j+4])+(image[i+4][j])+(image[i+4][j+1])+(image[i+4][j+2])+(image[i+4][j+3])+(image[i+4][j+4]))/25

End for

End for

for (i = 0 To i < SIZE)

for (j = 0 To j< SIZE)

assign image[i][j] = bimage[i][j]

End for

End for

End function

# main menu and calling the functions

declare rest

output ("ahlan ya user ya habibi ", char(3) ," :)")

loadImage() #loading the image

do: #starting page

declare op

output (" --- menu ---\n")

output ("1-Black & White Filter\n")

output ("2-Invert Filter\n")

output ("3-Merge Filter\n")

output("4-Flip Image\n")

output ("5-Darken and Lighten Image\n")

output ("6-Rotate Image\n")

output ("7-Detect Image Edges\n")

output ("8-Enlarge Image\n")

output("9-Shrink Image\n")

output ("10-Mirror 1/2 Image\n")

output ("11- Shuffle Image\n")

output ("12- Blur Image\n")

output ("13- Save the image to a file\n")

output ("14-exit\n")

output ("Please enter a choice from 1 to 14 >>> ")

input (op)

while (op <=0 OR op > 14)

output ("your input is not valid , please enter a choice in range [1,14] only\n")

input (op)

End while

If (op == 1) #1-bw filter

Call function bw\_filter()

Call function saveImage ()

else if (op == 2) #2-Invert Filter

Call function invert\_filter()

Call function saveImage ()

else if(op == 3) #3- Merge Filter

Call function merge\_filter()

Call function saveImage ()

else if(op == 4) #4- Flip Image

assign fch = 0

output ("choose a Flip 1- horizontally or 2-vertically ?\n")

output ("Please enter your choice >>> ")

input (fch)

while (fch <=0 OR fch > 2)

output ("your input is not valid , please enter 1 or 2 only\n")

input ( fch)

End while

if ( fch == 1 )

Call function flib\_filter()

Call function saveImage ()

else if ( fch == 2 )

Call function flib\_filter()

Call function rotate\_filter()

Call function rotate\_filter()

Call function saveImage ()

else if (op == 5) #5- Darken and Lighten Image

assign dch = 0

output ("choose a change 1- darken or 2-lighten ?\n")

output ("Please enter your choice >>> ")

input (dch)

while (dch <=0 OR dch > 2)

output ("your input is not valid , please enter 1 or 2 only\n")

input (dch)

End while

if (dch == 1 )

Call function d\_filter()

Call function saveImage()

else if (dch == 2)

Call function l\_filter()

Call function saveImage()

else if (op == 6) #6- Rotate Image

assign rch = 0

output (" Rotate (90), (180) or (270) degrees?\n")

ouputt ("Please enter the degrees >>> ")

input (rch)

while (rch != 90 and rch != 180 and rch != 270 )

output ("your input is not valid , please enter (90), (180) or (270) only\n" )

input (rch)

End while

if ( rch == 90 )

Call function rotate\_filter()

Call function saveImage ()

else if ( rch == 180 )

Call function rotate\_filter()

Call function saveImage ()

else if ( rch == 270 )

Call function rotate\_filter()

Call function rotate\_filter()

Call function rotate\_filter()

Call function saveImage ()

else if (op == 7) # 7- Detect Image Edges

Call function detect\_edges()

Call function saveImage ()

else if (op == 8) # 8- Enlarge Image

output ("Which quarter to enlarge 1, 2, 3 or 4 ?\n")

assign zch = 0

output<<"Please enter your choice >>> ")

input (zch)

while (zch <=0 ||zch > 4) {

output ("your input is not valid , please enter 1, 2, 3 or 4 only\n" )

input (zch)

End while

if (zch == 1)

zoom\_filter(0,0,128,128)

saveImage ()

else if (zch == 2)

zoom\_filter(0,128,128,256)

saveImage ()

else if (zch == 3){

zoom\_filter(128,0,256,128)

saveImage ()

else if (zch == 4)

zoom\_filter(128,128,256,256)

saveImage ()

else if (op == 9): # 9- Shrink Image

declare sch

output (" Shrink to a-(1/2) \*/\* b-(1/3) \*/\* c-(1/4)\n")

output("Please enter your choice >>> ")

input (sch)

while (sch != 'a' and sch != 'b' and sch != 'c' ) {

output("your input is not valid , please enter a or b or c only")

input (sch)

End while

if ( sch == 'a' )

shrink2\_filter()

saveImage ()

else if ( sch == 'b' )

shrink3\_filter()

saveImage ()

else if ( sch == 'c' ){

shrink4\_filter()

else if (op == 10): # a- Mirror 1/2 Image

output ("Mirror 1-left, 2-right, 3-upper, 4-down side?\n")

assign mch = 0

output("Please enter your choice >>> ")

input (mch)

while (mch <=0 OR mch > 4)

output ( "your input is not valid , please enter 1, 2, 3 or 4 only\n" )

input (mch)

End while

if (mch == 1)

mleft\_filter()

saveImage ()

else if (mch == 2)

mright\_filter()

saveImage ()

else if (mch == 3)

mup\_filter()

saveImage ()

else if (mch == 4)

downm\_filter()

saveImage ()

else if (op == 11): # b- Shuffle Image

shuffle\_filter()

saveImage ()

else if (op == 12): # c- Blur Image

blur\_filter()

saveImage ()

else if (op == 13) # Save the image to a file

Call function saveImage ()

else if (op == 14) #exit

output ("thanks for using our app\n")

else :

output (" sorry :( this filter is not available now , it will be available soon :)\n")

output ("Would you like to use the app again? (y/n)\n" )

input (rest)

while (rest != "y" and rest != "n" )

output ("your input is not valid , please enter y or n only <<<\n")

input (rest)

End while

while (rest == "y" )

output ("Thank you for using our app\n")

End while