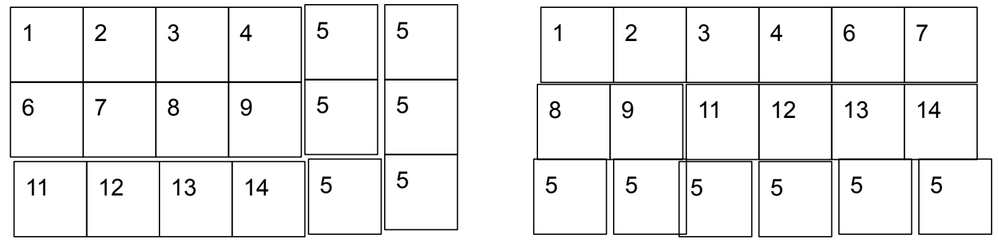
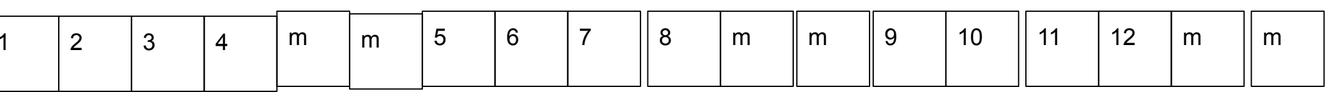
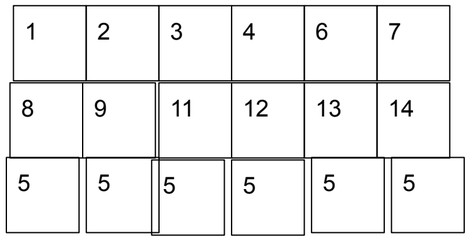


resin array






```
#include <iostream>
#include <cmath>
#include <map>
#include <cstdlib>

using namespace std;
void print_matrix(int **M, int row, int col)
{
  for(int i=0;i<row;i++)
  {
    for(int u=0;u<col;u++)
    {
      cout<<M[i][u]<<" ";
    }
    cout<<endl;
  }
  cout<<endl;
}

int** expansion_S(int **M,int **, int &row, int &col, int step, int exp_red, int fill_val)
{
  if(exp_red>0)
  {
    M=(int**)malloc(M.(row+step)*sizeof(int*));
    (*(int(*)*)malloc(*).(row*col+col*step)*sizeof(int));

    for(int i=0;i<row+step;i++)
    {
      M[i]=*(*)+*col;
    }

    for(int i=row;i<row+step;i++)
    {
      for(int u=0;u<col;u++)
      {
        M[i][u]=fill_val;
      }
    }
    row=row+step;
  }
  else if(exp_red<0 && row-step>=0)
  {
    M=(int**)malloc(M.(row-step)*sizeof(int*));
    (*(int(*)*)malloc(*).(row*col-col*step)*sizeof(int));

    for(int i=0;i<row-step;i++)
    {
      M[i]=*(*)+*col;
    }
    row=row-step;
  }
  //print_matrix(M,row,col);
  return M;
}

int** expansion_N(int **M,int **, int &row, int &col, int step, int exp_red, int fill_val)
{
  int t;
  if(exp_red>0)
  {
    M=(int**)malloc(M.(row+step)*sizeof(int*));
    (*(int(*)*)malloc(*).(row*col+col*step)*sizeof(int));

    for(int i=0;i<row+step;i++)
    {
      M[i]=*(*)+*col;
    }

    for(int i=row;i<row+step;i++)
    {
      for(int u=0;u<col;u++)
      {
        M[i][u]=fill_val;
      }
    }

    for(int i=(row+step)*col-col*step;i<(row+step)*col;i++)
    {
      for(int u=u;u<0;u--)
      {
        t=*(*)[u];
        (*)[u]=*(*)[u-1];
        (*)[u-1]=t;
      }
    }
    row=row+step;
  }
  else if(exp_red<0 && row-step>=0)
  {
    for(int i=row*step;i<0;i--)
    {
      for(int u=u*col;u<row-1;u++)
      {
        t=*(*)[u];
        (*)[u]=*(*)[u+1];
        (*)[u+1]=t;
      }
    }

    M=(int**)malloc(M.(row-step)*sizeof(int*));
    (*(int(*)*)malloc(*).(row*col-col*step)*sizeof(int));

    for(int i=0;i<row-step;i++)
    {
      M[i]=*(*)+*col;
    }
    row=row-step;
  }
  //print_matrix(M,row,col);
  return M;
}

int** expansion_E(int **M,int **, int &row, int &col, int step, int exp_red, int fill_val)
{
  int t;
  if(exp_red>0)
  {
    (*(*)+(int*)*realloc(*).((col+step)*row*sizeof(int));

    for(int i=0;i<row;i++)
    {
      M[i]=*(*)+*(col+step);
    }

    for(int i=col*row;i<(col+step)*row;i++)
    {
      (*)[i]=fill_val;
    }

    for(int i=0;i<row;i++)
    {
      for(int u=0;u<step;u++)
      {
        for(int j=col*row+u+7*step;j>(col+step)*(i+1)-step+u;j--)
        {
          t=*(*)[j];
          (*)[j]=*(*)[j-1];
          (*)[j-1]=t;
        }
      }
      col=col+step;
    }
    else if(exp_red<0 && col-step>=0)
    {
      for(int i=row-1;i>0;i--)
      {
        for(int u=0;u<step;u++)
        {
          for(int j=col-1-u;j>row*col-1;j++)
          {
            t=*(*)[j];
            (*)[j]=*(*)[j+1];
            (*)[j+1]=t;
          }
          //print_matrix(M, row, col);
        }
      }
      (*)=(int*)*realloc(*).((col-step)*row*sizeof(int));
      for(int i=0;i<row;i++)
      {
        M[i]=*(*)+*(col-step);
      }
      col=col-step;
    }
    return M;
  }

  int** expansion_W(int **M,int **, int &row, int &col, int step, int exp_red, int fill_val)
  {
    int t;
    if(exp_red>0)
    {
      (*)=(int*)*realloc(*).((col+step)*row*sizeof(int));

      for(int i=0;i<row;i++)
      {
        M[i]=*(*)+*(col+step);
      }
      for(int i=col*row;i<(col+step)*row;i++)
      {
        (*)[i]=fill_val;
      }
      for(int i=0;i<row;i++)
      {
        for(int u=0;u<step;u++)
        {
          for(int j=col*row+u+7*step;j>(col+step)*(i+1)-step+u;j--)
          {
            t=*(*)[j];
            (*)[j]=*(*)[j-1];
            (*)[j-1]=t;
          }
        }
        col=col+step;
      }
      else if(exp_red<0 && col-step>=0)
      {
        for(int i=row-1;i>0;i--)
        {
          for(int u=0;u<step;u++)
          {
            for(int j=col-1-u;j>row*col-1;j++)
            {
              t=*(*)[j];
              (*)[j]=*(*)[j+1];
              (*)[j+1]=t;
            }
            //print_matrix(M, row, col);
          }
        }
        (*)=(int*)*realloc(*).((col-step)*row*sizeof(int));
        for(int i=0;i<row;i++)
        {
          M[i]=*(*)+*(col-step);
        }
        col=col-step;
      }
      return M;
    }

    void life()
    {
      int row=3,col=5;
      int **M=new int*[row];
      int *+new int[row*col];
      for(int i=0;i<row;i++)
      {
        M[i]=++*col;
      }
      for(int i=0;i<row;i++)
      {
        for(int u=0;u<col;u++)
        {
          M[i][u]=0;
        }
      }
      //M=expansion_S(M,&r,row,col,6,1,4);
      print_matrix(M,row,col);
      M=expansion_S(M,&r,row,col,2,1,5);
      print_matrix(M,row,col);
      M=expansion_S(M,&r,row,col,2,1,9);
      print_matrix(M,row,col);*/

      //M=expansion_N(M,&r,row,col,4,1,7);
      //print_matrix(M,row,col);
      //M=expansion_N(M,&r,row,col,2,-1,7);
      //print_matrix(M,row,col);


      /*M=expansion_E(M,&r,row,col,3,1,1);
      print_matrix(M,row,col);
      M=expansion_E(M,&r,row,col,2,1,2);
      print_matrix(M,row,col);
      M=expansion_E(M,&r,row,col,2,1,3);
      print_matrix(M,row,col);
      M=expansion_E(M,&r,row,col,3,-1,1);
      print_matrix(M,row,col);*/


      M=expansion_W(M,&r,row,col,6,1,6);
      print_matrix(M,row,col);


      M=expansion_W(M,&r,row,col,3,-1,4);
      print_matrix(M,row,col);
    }

    int main()
    {
      life();
      return 0;
    }
```

start of life