Dynamic Shared Memory

Static Shared Memory

```
__global__ void staticReverse(int *d, int n)
{
  __shared__ int s[64];
  int t = threadIdx.x;
  int tr = n-t-1;
  s[t] = d[t];
  __syncthreads();
  d[t] = s[tr];
}

....

cudaMemcpy(d_d, a, n*sizeof(int), cudaMemcpyHostToDevice);
staticReverse<<<1,n>>>(d_d, n);
cudaMemcpy(d, d_d, n*sizeof(int), cudaMemcpyDeviceToHost);
```

Dynamic Shared Memory

```
__global__ void dynamicReverse(int *d, int n)
{
  extern __shared__ int s[];
  int t = threadIdx.x;
  int tr = n-t-1;
  s[t] = d[t];
  __syncthreads();
  d[t] = s[tr];
}

...

cudaMemcpy(d_d, a, n*sizeof(int), cudaMemcpyHostToDevice);
dynamicReverse<<<1,n,n*sizeof(int)>>>(d_d, n);
cudaMemcpy(d, d_d, n * sizeof(int), cudaMemcpyDeviceToHost);
```

Dynamic Shared Memory

```
__global__ void dynamicReverse(int *d, int n)
{
  extern __shared__ int s[];
  int t = threadIdx.x;
  int tr = n-t-1;
  s[t] = d[t];
  __syncthreads();
  d[t] = s[tr];
}

...

cudaMemcpy(d_d, a, n*sizeof(int), cudaMemcpyHostToDevice);
dynamicReverse<<<1,n,n*sizeof(int)>>>(d_d, n);
cudaMemcpy(d, d_d, n * sizeof(int), cudaMemcpyDeviceToHost);
```

1. Declare the shared array with `extern` qualifier
2. Declare the shared array as unsized
3. The invocation of the kernel has a third parameter that gives the size of the shared memory per block in bytes