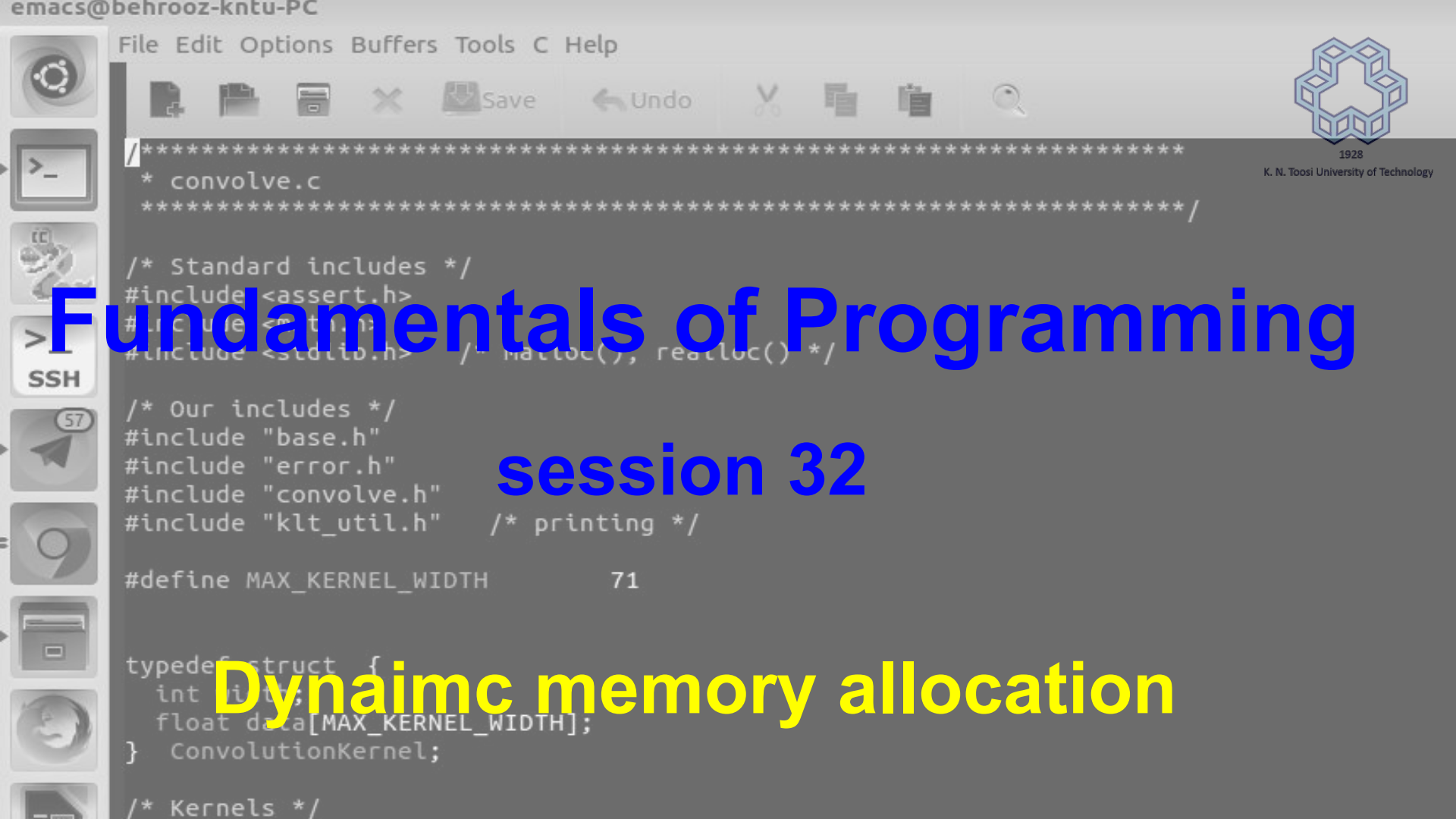
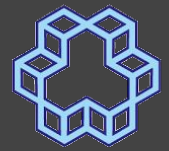
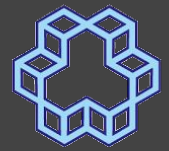

emacs@behrooz-kntu-PC
File Edit Options Buffers Tools C Help
Save
Undo
1928
K. N. Toosi University of Technology
/***
* convolve.c
***/
/* Standard includes */
#include <assert.h>
#include <math.h>
#include <stdlib.h> /* malloc(), realloc() */
/* Our includes */
#include "base.h"
#include "error.h"
#include "convolve.h"
#include "klt_util.h" /* printing */
#define MAX_KERNEL_WIDTH 71
typedef struct {
int width;
float data[MAX_KERNEL_WIDTH];
} ConvolutionKernel;
/* Kernels */
Fundamentals of Programming
session 32
Dynaimc memory allocation

Dynamic Memory Allocation with Stack




```c
void printArray(int a[], int n);

int main() {
  int n;

  scanf("%d", &n);

  int array[n];

  for (int i = 0; i < n; i++)
    array[i] = i;

  printArray(array,n);

  return 0;
}

void printArray(int a[], int n) {
```

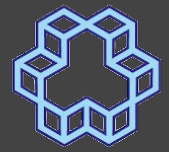
Dynamic Memory Allocation with Stack




```c
void printArray(int a[], int n);

int main() {
  int n;

  scanf("%d", &n);

  int array[n];

  for (int i = 0; i < n; i++)
    array[i] = i;

  printArray(array,n);

  return 0;
}

void printArray(int a[], int n) {
```

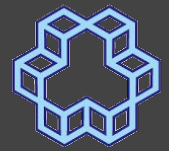
```
nasihatkon@kntu:code$ gcc dynamic1.c && ./a.out
16
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
```

Write as a function




```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

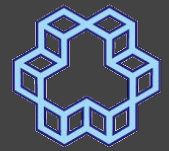
Write as a function




```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int a[n];

  for (int i = 0; i < n; i++)
    a[i] = i;

  return &a[0]; // or return a
}
```

Write as a function

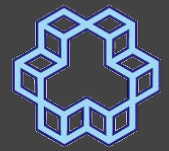



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int a[n];

  for (int i = 0; i < n; i++)
    a[i] = i;

  return &a[0]; // or return a
}
```

stack frame gets freed after returning from a function.

Write as a function

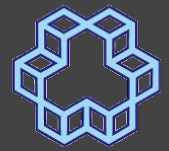



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int a[n];

  for (int i = 0; i < n; i++)
    a[i] = i;

  return &a[0]; // or return a
}
```

stack frame gets freed after returning from a function.

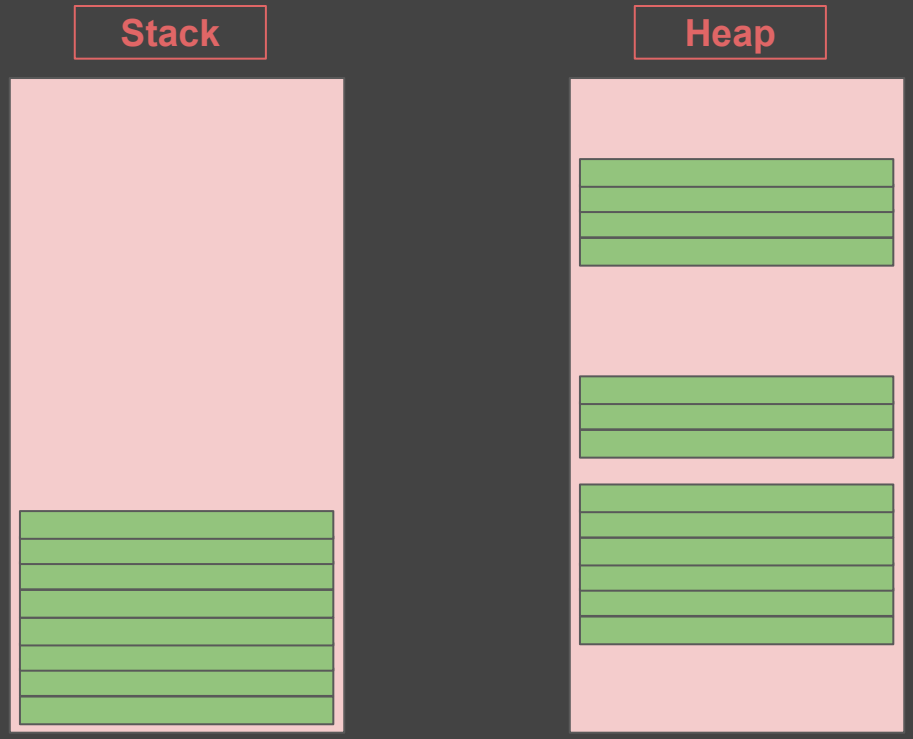
malloc: allocating memory from heap



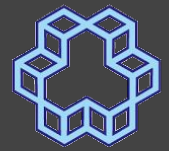

```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int *a;

  a = malloc(sizeof(int) * n);

  for (int i = 0; i < n; i++)
    a[i] = i;

  return a;
}
```

Stack vs. Heap



Write as a function

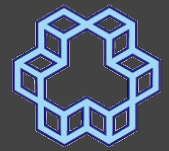



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int *a;

  a = malloc(sizeof(int) * n);

  for (int i = 0; i < n; i++)
    a[i] = i;

  return a;
}
```

```
nasihatkon@kntu:code$ gcc dynamic4.c && ./a.out
16
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
```

malloc: allocating memory from heap

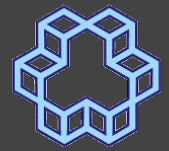



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  return 0;
}
```

```c
int *create_range(int n) {
  int *a;

  a = malloc(sizeof(int) * n);

  for (int i = 0; i < n; i++)
    a[i] = i;

  return a;
}
```

```
nasihatkon@kntu:code$ gcc dynamic4.c && ./a.out
16
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
```

Always free the allocated space from heap

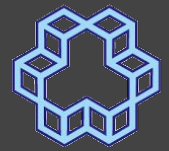



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  free(array);

  return 0;
}
```

```c
int *create_range(int n) {
  int *a;

  a = malloc(sizeof(int) * n);

  for (int i = 0; i < n; i++)
    a[i] = i;

  return a;
}
```

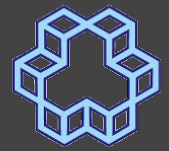
```
nasihatkon@kntu:code$ gcc dynamic4.c && ./a.out
16
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
```

pointer to void




```
MALLOC(3)                                  Linux Programmer's Manual

NAME
       malloc, free, calloc, realloc - allocate and free dynamic memory

SYNOPSIS
       #include <stdlib.h>

       void *malloc(size_t size);
       void free(void *ptr);
       void *calloc(size_t nmemb, size_t size);
       void *realloc(void *ptr, size_t size);
```

Always free the allocated space from heap

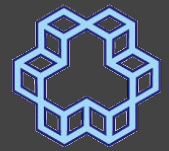



```c
int main() {
  int n;
  int *array;

  scanf("%d", &n);

  array = create_range(n);

  printArray(array,n);

  free(array);

  return 0;
}
```

```c
int *create_range(int n) {
  int *a;

  a = malloc(sizeof(int) * n);

  for (int i = 0; i < n; i++)
    a[i] = i;

  return a;
}
```

```
nasihatkon@kntu:code$ gcc dynamic4.c && ./a.out
16
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
```

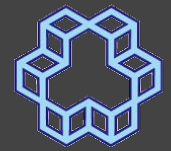
Remember: struct Employee




```c
struct Date {
  int  year;
  int  month;
  int  day;
};

struct Employee {
  char firstName[20];
  char lastName[20];
  struct Date DoB; // date of birth
  char gender;
  struct  Employee *supervisor;
};
```

```c
struct Employee amin_s, behnam_s, parham_s, mahdi_s;
amin_s    = createEmployee("Amin", "Parchami", 1378,6,7, 'M');
behnam_s = createEmployee("Behnam", "Beigi", 1340, 12, 25, 'M');
parham_s = createEmployee("Parham", "Parviz", 1390, 12, 30, 'M');
mahdi_s = createEmployee("Mahdi", "Forozan", 1380, 10, 5, 'M');

struct Employee *amin, *behnam, *parham, *mahdi;
amin = &amin_s;
behnam = &behnam_s;
mahdi = &mahdi_s;
parham = &parham_s;

amin->supervisor = NULL;
behnam->supervisor = amin;
mahdi->supervisor  = amin;
parham->supervisor = behnam;
```

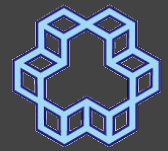
Remember: struct Employee




```c
struct Date {
  int  year;
  int  month;
  int  day;
};

struct Employee {
  char firstName[20];
  char lastName[20];
  struct Date DoB; // date of birth
  char gender;
  struct  Employee *supervisor;
};
```

```c
struct Employee createEmployee(char fName[], char lName[],
                               int birth_year, int birth_month,
                               int birth_day, char gender) {
  struct Employee s;

  strncpy(s.firstName, fName, 20);
  strncpy(s.lastName,  lName, 20);
  s.DoB.year  = birth_year;
  s.DoB.month = birth_month;
  s.DoB.day   = birth_day;
  s.gender = gender;

  return s;
}
```

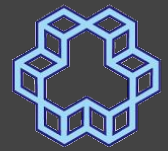
Rewrite using dynamic memory allocation




```
struct Employee createEmployee(char
                               int
                               int

  struct Employee s;

  strncpy(s.firstName, fName, 20);
  strncpy(s.lastName,  lName, 20);
  s.DoB.year  = birth_year;
  s.DoB.month = birth_month;
  s.DoB.day   = birth_day;
  s.gender = gender;

  return s;
}
```

```
struct Employee *createEmployee(char fName[], char lName[],
                                int birth_year, int birth_month,
                                int birth_day, char gender) {

  struct Employee *s = malloc(sizeof(struct Employee));

  strncpy(s->firstName, fName, 20);
  strncpy(s->lastName,  lName, 20);
  s->DoB.year  = birth_year;
  s->DoB.month = birth_month;
  s->DoB.day   = birth_day;
  s->gender = gender;

  return s;
}
```

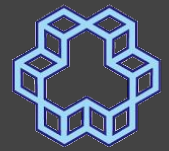
Rewrite using dynamic memory allocation




```
struct Employee createEmployee(char
                               int
                               int

  struct Employee s;

  strncpy(s.firstName, fName, 20);
  strncpy(s.lastName,  lName, 20);
  s.DoB.year  = birth_year;
  s.DoB.month = birth_month;
  s.DoB.day   = birth_day;
  s.gender = gender;

  return s;
}
```

```
struct Employee *createEmployee(char fName[], char lName[],
                                int birth_year, int birth_month,
                                int birth_day, char gender) {

  struct Employee *s = malloc(sizeof(struct Employee));

  strncpy(s->firstName, fName, 20);
  strncpy(s->lastName,  lName, 20);
  s->DoB.year  = birth_year;
  s->DoB.month = birth_month;
  s->DoB.day   = birth_day;
  s->gender = gender;

  return s;
}
```

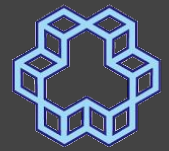
Remember: struct Employee




```
struct Employee amin_s, behnam
amin_s    = createEmployee("Am
behnam_s = createEmployee("Beh
parham_s = createEmployee("Par
mahdi_s = createEmployee("Mahd

struct Employee *amin, *behnam
amin = &amin_s;
behnam = &behnam_s;
mahdi = &mahdi_s;
parham = &parham_s;

amin->supervisor = NULL;
behnam->supervisor = amin;
mahdi->supervisor  = amin;
parham->supervisor = behnam;
```

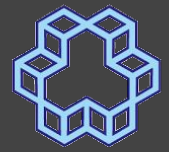
```
struct Employee *amin, *behnam, *parham, *mahdi;
amin   = createEmployee("Amin", "Parchami", 1378,6,7, 'M');
behnam = createEmployee("Behnam", "Beigi", 1340, 12, 25, 'M');
mahdi  = createEmployee("Parham", "Parviz", 1390, 12, 30, 'M');
parham = createEmployee("Mahdi", "Forozan", 1380, 10, 5, 'M');;

amin->supervisor = NULL;
behnam->supervisor = amin;
mahdi->supervisor  = amin;
parham->supervisor = behnam;

// finally free all dynamically allocated spaces
free(amin);
free(behnam);
free(mahdi);
free(parham);
```

Remember: struct Employee



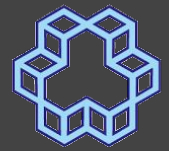

```
struct Employee *amin, *behnam, *parham, *mahdi;
amin   = createEmployee("Amin", "Parchami", 1378,6,7, 'M');
behnam = createEmployee("Behnam", "Beigi", 1340, 12, 25, 'M');
mahdi  = createEmployee("Parham", "Parviz", 1390, 12, 30, 'M');
parham = createEmployee("Mahdi", "Forozan", 1380, 10, 5, 'M');;

amin->supervisor = NULL;
behnam->supervisor = amin;
mahdi->supervisor  = amin;
parham->supervisor = behnam;

printEmployee(amin);
printEmployee(behnam->supervisor);
printEmployee(parham->supervisor);
printEmployee(parham->supervisor->supervisor);

free(amin);
free(behnam);
free(mahdi);
free(parham);
```

Practice: Linked List




```c
struct node {
   int value;
   struct node *next;
};
```

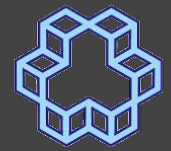
Practice: Linked List




```c
struct node {
  int value;
  struct node *next;
};
```

```c
int main() {
  int array[] = {10,20,30,40,50,60, 70};
  int n = sizeof(array)/sizeof(int);

  struct node *p;

  p = create_linked_list(array, n);

  print_linked_list(p);

  delete_linked_list(p);
}
```

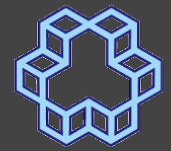
Practice: Linked List




```c
struct node {
  int value;
  struct node *next;
};
```

```c
struct node *create_linked_list(int a[], int n) {

  struct node *next = NULL;
  for (int i = n-1; i >= 0; i--) {
    struct node *q = malloc(sizeof(struct node));
    q->value = a[i];
    q->next  = next;
    next = q;
  }

  return next;
}
```

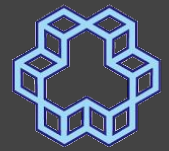
Practice: Linked List




```c
struct node {
  int value;
  struct node *next;
};
```

```c
void print_linked_list(struct node *p) {

  while (p != NULL) {
    printf("%d, ", p->value);
    p = p->next;
  }

  putchar('\n');

}
```

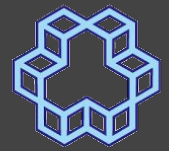
Practice: Linked List




```c
struct node {
  int value;
  struct node *next;
};
```

```c
void delete_linked_list(struct node *p) {

  while (p != NULL) {
    struct node *q;
    q = p;
    p = p->next;
    free(q);

  }

}
```

Practice: Linked List




```c
struct node {
  int value;
  struct node *next;
};
```

```c
int main() {
  int array[] = {10,20,30,40,50,60, 70};
  int n = sizeof(array)/sizeof(int);

  struct node *p;

  p = create_linked_list(array, n);

  print_linked_list(p);

  delete_linked_list(p);
}
```

```
nasihatkon@kntu:code$ gcc linked_list.c && ./a.out
10, 20, 30, 40, 50, 60, 70,
```