Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
2010-2011 Bahar Yarıyılı
Bilgisayar Sistemleri Lab. Dönem Sonu Sınavı

Tarih: 26 Mayıs 2011 Perşembe **Süre:** 120 dakika

Sadece 5 soruyu cevaplayınız.

1. Toplama ve faktöriyel işleçlerini içeren aritmetik ifadeler için bir CFG grameri yan tarafta verilmiştir. (20p)

CFG Grameri	Sınıflar
E → E "+" E	(Plus)
E → "!" "(" E ")"	(Fact)
E → ["0"-"9"]+	(Num)

a) Bu grameri JavaCC dilini (`.jj` uzantılı dosya formatında) kullanarak tanımlayınız ve tanımlamalarınıza nesne ağacını üreten Java yada C++ ifadelerini ekleyiniz.
b) Her bir gramer kuralı için ilgili sözdizim sınıfını yazınız.

2. Putty programı ile iki terminal penceresi (T1 veT2 terminalleri) açılıyor.
a) T1 terminalinden UNIX komutları girilerek icra edilirken, T2 terminalinden T1 terminalini gözetlemeyi sağlayacak komutu yazınız. (5p)
b) T1 terminalinde koşulan "`ls –l | grep myfork | wc | date`" komutunun çıktısı nedir? Bu komut izlendiğinde kaç adet süreç oluşturur? Bu komutun izlenmesinde oluşacak çocuk süreç sayısını veren komutu yazınız. (15p)

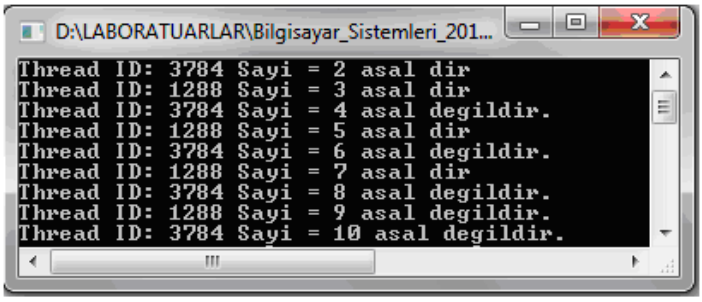
3. İstemciden aldığı küçük harf karakterlerini büyük harfe çevirerek tekrar istemciye geri gönderen Echo sunucusunu 4242 portunda dinlemede olacak şekilde kodlamanız istenmektedir. Örneğin, istemciden 'a' harfi gönderilmesi durumunda, sunucu 'A' ile cevap verecektir. Kodun yazılmasına yardımcı olması açısından, bazı fonksiyonlar ve parametreleri yanda ve aşağıda verilmiştir. Sunucu yazılımını gerçekleştirirken, hangi sunucu stratejisini kullandığınızı belirtiniz. (20p)

```
// constants: AF_INET,
PF_INET, SOCK_DGRAM,
SOCK_STREAM
struct sockaddr_in {
  short int sin_family;
  unsigned short int sin_port;
  struct in_addr sin_addr;
  unsigned char sin_zero [8];
}
```

```
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
int listen(int s, int backlog);
int socket(int domain, int type, int protocol);
```

4. İnfix notasyon kullanan bir hesap makinesinde giriş cümlelerinin sıfıra bölme hatası içermesi durumunda ekrana hatanın konumunun yazdırılması istenmektedir. Hata konumunu ekrana yazdırmak için kullanılacak, bölme işlemini tanımlayan üretim kuralına (production) ait semantik aksiyonun içeriğini Bison tanımlama diline uygun şekilde yazınız (Giriş dilindeki aritmetik ifadenin sıfıra bölme hatasını içermemesi durumunda ifadeye ait semantik değer normal bir şekilde hesaplanmalıdır). (20p)

5. Yan tarafta gösterilen ekran çıktısı oluşacak şekilde, 100'e kadar olan sayıların asal olup olmadığını kontrol eden thread'li bir uygulamayı **mutex semafor** senkronizasyon nesnesi kullanarak yazınız. (20p)



6. "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" verisini (40 tane a):

I	in dictionary?	new entry	output
a	Y		
aa	N	.	.
.	.	.	.
.	.	.	.
.	.	.	.

a) LZW yöntemine göre kodlayarak yandaki tabloda gösterilen formatta yazınız.
b) 16 bit ile kodlandığında (EOF karakterini de dikkate alarak) veri sıkıştırma oranını (output/input) %. . . olarak hesaplayınız. (20p)

7. **a)** Bellekte 256 byte'lık bir alan(segment) tahsis edip, tahsis edilen bu alana 2 basamaklı herhangi bir sayı yazan C kodunu yazınız (Ayrılan segment 0x7500000 adresi ile adreslenecektir.). (16p)
Hatırlatma: `shmget( .. , .. , .. )` --> Bellekte alan ayırmak için kullanılır.
`shmat( .. , .. , .. )` --> Segment adreslemede kullanılır.
b) Yazdığınız C kodunun derlenip çalıştırıldığında bellekte garbage oluşturmaması için eklenmesi gereken kod nedir? (4p)

2010-2011 Bilgisayar Sistemleri Laboratuarı Final Sınavı Cevapları

***** Cevap 1 *****

a) JavaCC tanımlaması

```
PARSER_BEGIN(Parser)
public class Parser { }
PARSER_END(Parser)

SKIP : { " " | "\t" | "\r" }
 TOKEN : { < NUMBER : (["0"-"9"])+ ("."
   (["0"-"9"])+)? > }
TOKEN : { < EOL : "\n" > }
TOKEN : /* OPERATORS */
{  < PLUS: "+" >
 | < FACT: "!" >
 | < LPR: "(" >
 | < RPR: ")" >
}
Exp parse() : {
  Exp a;
}
{
  a = plus() (<EOL>) { return a; }
}
Exp plus() : { Exp a, b; }
{ a = fact() ( <PLUS> b=fact()
    { a = new Plus(a, b); } )*
  { return a; }
}
```

```
Exp fact() : { Exp a; }
{ t=<NUMBER>
    { return new Num(
          Double.parseDouble(t.image));
    }
| <FACT> <LPR> a=plus() <RPR>
    { retun (new Fact(a); }
}
```

b) Sözdizim sınıfları

```
abstract class Exp { }

class Plus extends Exp {
  public Exp exp1, exp2;

  public Plus(Exp e1, Exp e2) {
    exp1 = e1; exp2 = e2;
  }
}

class Fact extends Exp {
  public Exp exp;

  public Fact(Exp e) {
    exp = e;
  }
}
```

***** Cevap 2 *****

a) T2 terminali komut satırına "`truss -p` *SH1'nin pid*" yazılır. SH1 nin pid sini öğrenmek içinde T1 terminalinde "`echo $$`" komutu girilir.

b) `date` komutunun çıktısını verir. Dört child bir parent olmak üzere beş adet süreç oluşturur. T2 terminali T1 terminalini gözetleyecek şekilde "`truss -o` *dosya* `-f -p SH1`" çalıştırılır. Ardından T2 terminalinde "`cat` *dosya* `| grep child | wc -l`" komutu child sayısını verir.

***** Cevap 3 *****

```
#define thePort 4242
int main() {
  int soc, ns, result ;
  /* 1 mark each for defining peer, self;
     2 marks for setting port number */
  struct sockaddr_in self = {AF_INET,
     htons(thePort)};
  struct sockaddr_in peer = {AF_INET};
  char c ;
  soc = socket(AF_INET, SOCK_STREAM, 0);
  /* 2 marks */
  if (soc == -1) {
    perror("Socket");
    exit(1); }
  /* 2 marks for bind() */
  result = bind(soc,
    (struct sockaddr)&self, sizeof(self));
  if (result == -1) {
    perror("Bind");
    exit(1); }
  if (listen(soc, 1) == -1) {/* 2 marks */
    perror("Listen");
    exit(1); }
  for (;;) { /* 2 marks for accept() */
    ns = accept(soc,
          (struct sockaddr_in )&peer,
          sizeof(peer));
    if (ns = -1) {
       perror("Accept");
       exit(1);
    }
    if (fork() == 0) {
       /* 1 mark (read) */
       while (read(ns, &c, 1)) {
         c = toupper(c); /* 1 mark */
         write(ns, &c, 1); /* 1 mark */
       }
       exit(0);
    }
  }
  return 0;
}
```

***** Cevap 4 *****

```
exp : NUM { $$ = $1; }
| exp '+' exp { $$ = $1 + $3; }
| exp '-' exp { $$ = $1 - $3; }
| exp '*' exp { $$ = $1 * $3; }
| exp '/' exp {
  if ($3)
  else
  {
    $$ = 1;
    fprintf (stderr,
       "%d.%d-%d.%d: division by zero",
       @3.first_line, @3.first_column,
```

```
    $$ = $1 / $3;
```

```
      @3.last_line, @3.last_column);
  }
}
| '-' exp %prec NEG { $$ = -$2; }
| exp '^' exp { $$ = pow ($
```

******* Cevap 5 *******

```
int asal(int n) {
  for(int i = 2; i <
      (int)(sqrt((float)n) + 1.0) ; i++) {
    if (n % i == 0) return 0;}
      return 1;
}

unsigned int __stdcall mythread(void*) {
  char* s;
  while (sayi < 100) {
    WaitForSingleObject(mutex, INFINITE);
    Sleep(1);
    sayi++;
    s = "degildir.";
    if(asal(sayi)) s = "dir";
      printf("Thread ID: %d Sayi = %d asal
%s\n",GetCurrentThreadId(), sayi, s);
    ReleaseMutex(mutex);
  }
  return 0;
}
```

```
void main() {
  HANDLE  a,b;
  mutex = CreateMutex(0, 0, 0);
  a = (HANDLE)_beginthreadex(0, 0,
        &mythread, (void*)0, 0, 0);
  b = (HANDLE)_beginthreadex(0, 0,
        &mythread, (void*)1, 0, 0);
  WaitForSingleObject(a, INFINITE);
  WaitForSingleObject(b, INFINITE);
  CloseHandle(a);
  CloseHandle(b);
  CloseHandle(mutex);
  getchar();
}
```

******* Cevap 6 *******

I	in dictionary ?	new entry	output
a	Y		
aa	N	256-aa	97(a)
aa	Y		
aaa	N	257-aaa	256(aa)
a	Y		
aa	Y		
aaa	Y		
aaaa	N	258-aaaa	257(aaa)
a	Y		
aa	Y		
aaa	Y		
aaaa	Y		
aaaaa	N	259-aaaaa	258(aaaa)
a	Y		
aa	Y		
aaa	Y		
aaaa	Y		
aaaaa	Y		
aaaaaa	N	260-aaaaaa	259(aaaaa)
a	Y		
aa	Y		
aaa	Y		
aaaa	Y		
aaaaa	Y		
aaaaaa	Y		
aaaaaaa	N	261-aaaaaaa	260(aaaaaa)
a	Y		

aa	Y		
aaa	Y		
aaaa	Y		
aaaaa	Y		
aaaaaa	Y		
aaaaaaa	Y		
aaaaaaaa	N	262-aaaaaaaa	261(aaaaaaa)
a	Y		
aa	Y		
aaa	Y		
aaaa	Y		
aaaaa	Y		
aaaaaa	Y		
aaaaaaa	Y		
aaaaaaaa	Y		
aaaaaaaaa	N	263-aaaaaaaaa	262(aaaaaaaa)
a	Y		
aa	Y		
aaa	Y		
aaaa	Y		
aaaa,EOF	N		258(aaaa)

******* Cevap 7 *******

a)

```
#include <stdio.h>
#include <sys/shm.h>
int main () {
  int segment_id;
  int rastgele;
  char* shared_memory;
  int segment_size;
  //256 byte
  const int shared_segment_size = 0x100;
segment_id = shmget (IPC_PRIVATE,
     shared_segment_size, IPC_CREAT |
     IPC_EXCL | S_IRUSR | S_IWUSR);
  shared_memory = (char*)
    shmat (segment_id,0x7500000, 0);
  rastgele = 10 + (int) (
    90 * rand () / (RAND_MAX + 1.0));
  spintf(shared_memory,"%d",rastgele);
        }
```

b) shmctl(segment_id,IPC_RMID,0);

Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
2011-2012 Bahar Yarıyılı
Bilgisayar Sistemleri Lab. Dönem Sonu Sınavı

Tarih: 1 Haziran 2012 Cuma **Süre:** 120 dakika

Sadece 5 soruyu cevaplayınız.

1. Bir ayrıştırıcı (parser) yazılarak, genel biçimi aşağıda verilen `f(x)` fonksiyonu için nesne ağacı üretilmek isteniyor. Fonksiyonda `1+x` terimlerinin sayısına herhangi bir sınırlama getirilmemiştir. (20p)

```
1+x(1+x(1+x(...(1+x)...)))
```

a) Bu fonksiyon üzerinde çalışacak ayrıştırıcıyı JavaCC dilini (`.jj` uzantılı dosya formatında) kullanarak tanımlayınız ve tanımlamalarınıza nesne ağacını üreten Java yada C++ ifadelerini ekleyiniz.

b) Nesne ağacı için gereken sözdizim sınıf(lar)ını tanımlayınız.

2. Putty programı ile T1 ve T2 terminal pencereleri açılıyor ve T2'den T1 terminali gözetlenmek isteniyor.

a) T1 terminalinde "**who | grep a | wc**" komutu koşulduğunda kaç çocuk süreç oluşur? T2 terminalinde bu çocuk süreçlerin de gözetlenmesini sağlayarak, bu komutun icrasında oluşan çocuk süreç sayısını veren komutu yazınız. (10p)

b) /proc dosya sisteminde bulunan **prempтyset**, **praddset** fonksiyonlarının görevini açıklayınız.(10p)

3. **a)** Eş zamanlı bir sunucu (Concurrent Server), aynı anda birden fazla istemciye hizmet verebilmek için farklı stratejiler kullanmaktadır. Gerek fork() sistem çağrısına dayanan, gerekse thread'leri kullanan stratejilerde istemciye hizmet verildikten sonra close() sistem çağrısı ile, o anki istemci ile ilişkilendirilmiş olan soket kapatılmaktadır. Böyle bir programlama yaklaşımının kullanılma sebebi nedir? (10p)

b) TCP Sunucu programın, kullanması gereken sistem çağrılarını akış diyagramını vererek kullanım sebepleri ile beraber açıklayınız. (10p)

4. Infix notasyonu kullanan hesap makinesi programına kök operatörünün (*root*) eklenmesi istenmektedir. *root* operatörü yanda gösterildiği gibi yalnızca 2 farklı şekilde çalışmaktadır. Hesap makinesinin root operatörüne ilave olarak `+`, `-`, `*` ve `/` operatörlerini desteklediği varsayıldığında operatör önceliklerinin en yüksek öncelikliden en düşük öncelikliye doğru sırasıyla `*`, `/`, *root* (küpkök durumu), *root* (karekök durumu), `+`, `-` şeklinde değişmesi gerekmektedir.

```
root 25    // kare-kök
5.0
64 root 3  // küp-kök
4.0
```

Not: Operatör birleşme özellikleri (associativity) istenildiği şekilde seçilebilir. Gramer kurallarındaki aritmetik ifadeleri temsil etmek için *exp* non-terminal sembolü kullanılabilir.

a) root operatör önceliğinin nasıl tanımlanması gerektiğini gösteriniz. (10p)

b) root operatörüne ait gramer kurallarını belirterek karşılık düşen semantik aksiyonları yazınız. (10p)

5. Çoklu threadli bir uygulamada, a[n] ve b[n] olmak üzere 2 tane global tanımlanan dizi bulunmaktadır. Programdaki thread sayısı dizi boyutları (n) kadardır. Her bir thread yan taraftaki kod parçasını icra etmektedir. Fonksiyon **f1()**, iki tane integer değer alıp geriye tek integer değer döndürürken, **f2()** tek bir integer değer alıp tek integer değer döndürmektedir. f1() ve f2() fonksiyonları herhangi bir global değişken kullanmıyorlar.

```
Thread_Tj
{
  ...
  a[j]=f1(a[j-1],a[j]);
  b[j]=f2(a[j]);
  ...
}
```

Bu uygulamada `Thread_Tj`, başka bir threadin hesapladığı a[j-1] değerini almaktadır ve a[j] değeri `Thread_Tj` fonksiyonunda *f1()* yardımıyla hesaplanmaktadır. b[j] değerleri de, hesaplanan a[j] değerini parametre alan *f2()* fonksiyonunda güncellenmektedir.

Tüm threadler koştuktan sonra, dizilerin son değerlerini ekran çıktısı olarak veren uygulamayı ***kritik bölge nesnesini*** kullanarak gerçekleyiniz. (20p)

6. **a)** LZW yöntemiyle veri sıkıştırma yapılırken dictionary tablosunun boyu arttıkça optimum BITS değeri öncesi/sonrası sıkıştırma süresinin değişimini **ARTAR/AZALIR**; veri sıkıştırma oranını ise **İYİLEŞİR /KÖTÜLEŞİR** olarak yandaki tabloya yazınız. (10p)

b) LZW yöntemiyle veri sıkıştırmada optimum BITS değerinin bulunması için bir yöntem öneriniz. (10p)

		Optimum BITS **Öncesi**	Optimum BITS **Sonrası**
Dictionary Tablosunu Boyu Arttıkça	Veri Sıkıştırma **Süresi**		
	Veri Sıkıştırma **Oranı**		

7. Shared memory tekniği kullanılarak süreçler haberleştirilmek isteniyor. Bunun için bellekte 4096 byte'lık alan açılıp 0x200000 adresi ile adreslenecektir. Süreçlerden biri komut satırından parametre olarak aldığı double değerde bir sayının tam kısmını belleğe yazarken, diğer süreç bu değeri okuyup ekrana bastıracaktır. İlgili C kodunu yazınız. (20p)

2011-2012 Bilgisayar Sistemleri Laboratuarı Final Sınavı Cevapları

***** Cevap 1 *****

a) JavaCC tanımlaması

```
TOKEN :
{
    < X: "x" >
  | < ONE: "1" >
  | < LPAR: "(" >
  | < RPAR: ")" >
  | < PLUS: "+" >
  | < EOL: "\n" >
}

Exp parse() : {
  Exp e;
}
{
  e = exp() <EOL> { return e; }
}

Exp exp() : {
  Exp v, t;
}
{
  <ONE> <PLUS> <X> { v=new Var(); } (
    <LPAR>  t=exp()  <RPAR>  {  v=new
   Times(v, t); }
  )? { return new Plus(1, v); }
}
```

```
abstract class Exp { }

class Plus extends Exp {
  public int n;
  public Exp exp;

  public Plus(int x, Exp e) {
    n = x; exp = e;
  }
}

class Times extends Exp {
  public Exp exp1, exp2;

  public Times(Exp e1, Exp e2) {
    exp1 = e1; exp2 = e2;
  }
}

class Var extends Exp
{
  public Var() { }
}
```

***** Cevap 2 *****

```
T1: echo $$
T2: truss –o dosya –f –p T1PID
T2: grep child dosya | wc-l
```

***** Cevap 3 *****

a) Why is it a good idea to close() each socket when you are done with it?

Besides being good programming to clean up after yourself, you can potentially run out of file descriptors in a program if you do not close sockets. This is particularly true for servers that must handle multiple client requests … without closing, you eventually lose the ability to accept new client requests. Furthermore, closing a socket also allows the port number and other resources allocated to the socket to be potentially freed sooner by the operating system.

b) TCP Sunucu programın, kullanması gereken sistem çağrılarını akış diyagramını vererek kullanım sebepleri ile beraber açıklayınız.

Socket(), bind(), listen(), accept(), read() ve close()

***** Cevap 4 *****

a)

```
left '+'  '-'
left 'root'  /*karekök durumu*/
left ROOT /*küpkök durumu*/
left '*'  '/'

//Gramer kuralı
exp: 'root' exp %prec ROOT
```

b)

```
exp: exp root exp  {$$ =
pow($1,1/$3);} /*küpkök gramer
kuralı*/
exp: 'root' exp %prec ROOT {$$ =
sqrt($2)} /*karekök gramer kuralı*/
}
```

***** Cevap 5 *****

```
int Thread(void) {
  ...
  EnterCriticalSection(&cs);
```

```
...
HANDLE aThread[threadsayac];
DWORD ThreadID;
```

```
if(j<(n-1)) {
  a[j+1]=f1(a[j],a[j+1]);
  b[j]=f2(a[j+1]);
  j++;
}

  LeaveCriticalSection( &cs );
  ...

  return 0;
}
```

```
CRITICAL_SECTION cs;
int a[n],b[n];
...

void main (void)  {
  InitializeCriticalSection( &cs );

  for(int i=0; i < threadsayac; i++ )
  {
    aThread[i] = CreateThread(NULL,0,
      (LPTHREAD_START_ROUTINE)Thread,
      NULL,0,&ThreadID);
     ...
  }

  WaitForMultipleObjects(THREADCOUNT,
    aThread, TRUE, INFINITE);
  dizidegerlerigoster();

  for(int i=0; i < threadsayac; i++ )
    CloseHandle(aThread[i]);
  DeleteCriticalSection(&cs);
  ...
   return 0;
}
```

***** **Cevap 6** *****

a)

		Optimum BITS **Öncesi**	Optimum BITS **Sonrası**
Dictionary Tablosunun Boyu Arttıkça	Veri Sıkıştırma **Süresi**	ARTAR	AZALIR
	Veri Sıkıştırma **Oranı**	İYİLEŞİR	KÖTÜLEŞİR

b) 12 gibi düşük bir BITS değerinden başlanılarak veri LZW kodlanırken dictionary tablosunun dolup/dolmadığı test edilir. Verinin tamamı kodlanmadan tablo dolduğu müddetçe BITS değeri 1 arttırılır. Tabloda ilk boşluğa rastlanılan veya en son dolduğu (1 eksiği) BITS değeri optimum BITS değeri olarak alınır.

***** **Cevap 7** *****

```
#include "stdio.h"
#include "sys/shm.h"
#include "sys/stat.h"

int main(int argc, char* const argv[]) {
  double num;
  int intpart;
  const int shared_seg_size=0x1000;
  int segment_id;
  char* shared_memory;

  num = argotpart=(int) num;
  segment_id = shmget(...,shared_seg_size,...);
  shared_memory = shmat (segment_id,0x200000,0);
  sprintf(shared_memory, "%d", intpart);
  printf("%d", shared_memory);
  shmtdt(shared_memory);
  shmctl(segment_id,IPC_RMID,...);
}
```

Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
2012-2013 Bahar Yarıyılı
Bilgisayar Sistemleri Lab. Dönem Sonu Sınavı

Tarih: 31 Mayıs 2013 Cuma **Süre:** 120 dakika

Sadece 5 soruyu cevaplayınız.

1. C ile aynı sözdizimine sahip bir programlama dilinde tamsayı parametreleri verilerek çağrılan ve gövdesi tanımlanmayan fonksiyonlar kendisine geçilen verilerin toplamını (parametresi yoksa `0`) hesaplamaktadır. Fonksiyon isimleri küçük harflerden oluşmaktadır ve parametrelerin sayısında herhangi bir sınırlama yoktur. Fonksiyonların birkaç örneği aşağıda gösterilmiştir.

```
ff();      fn(12);      fs(3,0,-2);      foo(1,2,3,4,5);
```

a) Bu çeşit fonksiyonlar için bir ayrıştırıcıyı, JavaCC dilini (`.jj` uzantılı dosya formatında) kullanarak ve nesne ağacını üreten Java yada C++ ifadelerini ekleyerek tanımlayınız. (10p)

b) Bu çeşit fonksiyonlar için bir değerlendiriciyi, `intanceof` işlecini kullanarak yazınız. (10p)

2. Putty programı ile T1 ve T2 terminal pencereleri açılıyor ve T2'den T1 terminali gözetlenmek isteniyor.

a) T1 terminalinde "**ls | grep ls | wc -l**" komutu koşulduğunda kaç çocuk süreç oluşur? T2 terminalinde bu çocuk süreçlerin sayısını veren ve gözetlenmesini sağlayan komutları yazınız. (10p)

b) `ls > LS` komutunun yapabileceği sistem çağrılarını argümanlarıyla birlikte sırasıyla yazınız. Yapılan toplam sistem çağrısı sayısı belirlenebilir mi? Belirlenebilirse kaç adet sistem çağrısı yapmıştır? (10p)

3. **a)** LZW ile veri sıkıştırmada optimum BITS değerinin bulunması için dictionary tablosu ile ilişkili bir yöntem öneriniz. (5p)

b) Dictionary tablosunun boyu arttıkça optimum BITS değeri öncesi/sonrası veri sıkıştırma oranı **İYİLEŞİR/KÖTÜLEŞİR** olarak aşağıdaki tabloya yazınız ve nedenini hem öncesi hem de sonrası için açıklayınız. (15p)

	Optimum BITS **Öncesi**	Optimum BITS **Sonrası**
Tablo Boyu Arttıkça Veri Sıkıştırma Oranı		

4. **a)** Eş zamanlı bir sunucu (Concurrent Server), aynı anda birden fazla istemciye hizmet verebilmek için farklı stratejiler kullanmaktadır. Hem `fork()` sistem çağrısına dayanan, hem de thread'leri kullanan stratejilerde istemciye hizmet verilmektedir. Bu stratejilerin hangi durumlarda avantaj ya da dezavantajları vardır birbirleriyle karşılaştırarak ve örnek vererek açıklayınız. (10p)

b) TCP Sunucu programın, kullanması gereken sistem çağrılarını akış diyagramını vererek kullanım sebepleri ile beraber açıklayınız. (10p)

5. Infix notasyonu kullanan hesap makinesi programına myOP operatörünün eklenmesi istenmektedir. myOP operatörü 2 farklı şekilde çalışmaktadır. "myOp Sayı" kullanımı Sayı*Sayı değerini üretirken, "Sayı1 myOp Sayı2" kullanımı Sayı1*Sayı2 değerini üretmelidir. Hesap makinesinin myOp operatörüne ilave olarak * ve / operatörlerini desteklediği varsayıldığında operatör önceliklerinin en yüksek öncelikliden en düşük öncelikliye doğru sırasıyla *, /, myOp (tekli operand durumu) ve myOp (ikili operand durumu) şeklinde değişmesi gerekmektedir.

Not: Operatör birleşme özellikleri (associativity) istenildiği şekilde seçilebilir. Gramer kurallarındaki aritmetik ifadeleri temsil etmek için *exp* terminal-olmayan sembolü kullanılabilir.

a) myOp operatör önceliğinin nasıl tanımlanması gerektiğini gösteriniz. (10p)

b) myOp operatörüne ait gramer kurallarını belirterek karşılık düşen semantik aksiyonları yazınız. (10p)

6. Haritalanmış bellek yöntemi ile <sistem_lab> adlı dosya içerisine rastgele bir değer yazan uygulama kodları aşağıda verilmiştir. Haritalanmış bellek yöntemini kullanarak, <sistem_lab> dosyasındaki sayısal değeri okuyan ve okunan değere 5 ekleyip tekrar aynı dosya içerisine kaydeden uygulamayı kodlayınız. (20p)

```
int main () {
  int fd;  void* file_memory;
  fd = open ("sistem_lab", O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);
  lseek (fd, 0x100+1, SEEK_SET);
  write (fd, "", 1);
  lseek (fd,0, SEEK_SET);
  file_memory = mmap (0, 0x100, PROT_WRITE, MAP_SHARED, fd,0);
  close (fd);
  sprintf((char*) file_memory, "%d\n", rastgele_deger_uret(-100,100));
  munmap (file_memory, FILE_LENGTH);
  return 0; }
```

7. *N* tane yolcunun ve bir tane servis aracının bulunduğu bir taşımacılık sistemi olsun. Yolcular, *X* yolcu kapasiteli araca binmek için kuyrukta beklemektedir (X<N). Araç, tamamen dolduktan sonra hareket etmektedir. Servise çıkan araç, hareket halindeyken araçtan yolcu inmemekte ve araca yolcu binmemektedir. Servis tamamlandığında yolcular araçtan teker teker inecektir ve yolcular inene kadar araca yolcu binemeyecektir. Aracın toplam servis sayısı *Y* kadardır. Bu sistemi modelleyen, **Yolcu()** ve **Araç()** thread fonksiyonlarını pseudo-code şeklinde, ***sadece semafor senkronizasyon nesnesini*** kullanarak yazınız. (20p)

2012-2013 Bilgisayar Sistemleri Laboratuarı Final Sınavı Cevapları

***** Cevap 1 *****

a) JavaCC tanımlaması

func.jj

```
options {
    STATIC = true; // make parser
methods static
//    DEBUG_PARSER = true;
    LOOKAHEAD = 1;
}

PARSER_BEGIN(Func)
public class Func {
}
PARSER_END(Func)

SKIP : { " " | "\t" | "\r" }

TOKEN : /* OPERATORS */
{
    < #DIGIT: ["0"-"9"] >
  | < NUM: (<DIGIT>)+ >
  | < EOL : "\n" >
  | < FNAME : ["a"-"z","A"-"Z"](["a"-
"z","A"-"Z"]|["0"-"9"])* >
}

Exp parse() : {
  Exp fn;
}
{
  fn=Expression()(<EOF> | <EOL>) {
return fn; }
}

Exp Expression() : {
  Token t;
  Exp exp[];
}
{
    t=<FNAME> "(" exp=Form() ")" {
return new Function(t.image, exp); }
  | t=<NUM> { return new Num(t.image);
}
}

Exp[] Form() : {
  Exp exp[] = new Exp[50];
}
{
  (Form2(exp, 0) { return exp; })? {
return null; }
}

void Form2(Exp exp[], int ind) : {
  Exp e;
}
{
  e=Expression() { exp[ind]=e;
exp[ind+1]=null; } ("," Form2(exp,
ind+1))?
}
```

AST.java

```
class Exp {  }

class Num extends Exp {
  String n;
  public Num(String s) {
      n = s;
  }
}

class Function extends Exp{
  String name;
  Exp exp[];
  public Function(String n, Exp e[])
{
      name = n;
      exp = e;
  }
}
```

***** Cevap 2 *****

```
T1: echo $$
T2: truss –o dosya –f –p T1PID
T2: grep child dosya | wc-l
```

Bir komutun yaptığı sistem çağrılarının sayısı, onun kaynak kodundaki görünür ifadelerle sınırlı değildir.

Kütüphane dosyalarının açılıp okunması ve komutu koşacak sürecin yönetimi gibi gereksinmeleri karşılamak için gerekli bazı kod parçaları da komutun derlenmesi esnasında icra edilebilir koda dahil edilir. Bu kod parçaları birçok sistem çağrısının yapılmasına ihtiyaç duyar.

***** Cevap 3 *****

a) 12 gibi düşük bir BITS değerinden başlanılarak veri LZW kodlanırken dictionary tablosunun dolup/dolmadığı test edilir. Verinin tamamı kodlanmadan tablo dolduğu müddetçe BITS değeri 1 arttırılır. Tabloda ilk boşluğa rastlanılan veya en son dolduğu (1 eksiği) BITS değeri optimum BITS değeri olarak alınır.

b) Optimum BITS öncesi verilerin tamamı kodlanamadan tablo sürekli dolduğundan her yeni BITS değerinde öncekine nazaran daha fazla yeni kod eklenir ve bu da veri sıkıştırma oranını iyi yönde etkiler.
Optimum BITS sonrası tablo mevcut dosyanın yeni bir kod eklenmeksizin kodlanması için yeterli olduğundan

	Optimum BITS Öncesi	Optimum BITS Sonrası
Tablo Boyu Arttıkça Veri Sıkıştırma Oranı	İYİLEŞİR	KÖTÜLEŞİR

yani artık tabloya yeni bir kod eklenmeyeceğinden herbir BIT artışında kodların herbiri 1 BIT fazla kodlanacak ve bu da veri sıkıştırma oranını kötü yönde etkileyecektir.

******* Cevap 4 *******

a) `fork()` ⇒ yeni çocuk süreç oluşur. Çocuk süreç farklı bir bellek alanı kullanır. İstenildiği zaman `kill` komutuyla durdurulabilir.

b) socket(), bind(), listen(), accept(), read() ve close()
socket(), connect(), send(), recv(), close()

******* Cevap 5 *******

a)
```
left 'myOp'  /*ikili operand durumu*/
left MYOP /*tekli operand durumu*/
left '*' '/'

//Gramer kuralı
exp: 'myOp' exp %prec MYOP
```

b)
```
exp: exp 'myOp' exp  {$$ = $1*$3;}
     /*ikili operand durumu için gramer kuralı*/
exp: 'myOp' exp %prec MYOP {$$ = pow($2,2)}
     /*tekli operand durumu için gramer kuralı*/
```

******* Cevap 6 *******

```
int main (int argc, char* const argv[]) {
  int fd;
  void* file_memory;
  int okunan_deger;

  fd = open ("sistem_lab", O_RDWR, S_IRUSR | S_IWUSR);
  file_memory = mmap (0, FILE_LENGTH, PROT_READ | PROT_WRITE,
                  MAP_SHARED, fd, 0);
  close (fd);

  sscanf (file_memory, "%d", &okunan_deger;);
  sprintf ((char*) file_memory, "%d\n ", okunan_deger+5);
  munmap (file_memory, FILE_LENGTH);

  return 0;
}
```

******* Cevap 7 *******

```
Arac() {
 for(int i=0;i<Y;i++) {
  artır(yolcu_sırası);  //en az X tane
yolcu sırada bulunmalı
 azalt(yolcu_bindir); //başlangıç
değeri X
 servis();
 for(int i=0;i<X;i++) {
  artır(arac_servis); //yolcu
indirilmesi
  azalt(yolcu_indir); //her bir
yolcunun inişinin tamamlanması
beklenecektir.
}
}
}
```

```
Semafor  yolcu_sırası
Semafor  yolcu_bindir
Semafor  arac_servis
Semafor  yolcu_indir

Yolcu() {
 while(true) {
  azalt(yolcu_sırası);
  artır(yolcu_bindir);
  servis2();
  azalt(arac_servis);
  artır(yolcu_indir);
 }
}
```

Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
2013-2014 Bahar Yarıyılı
Bilgisayar Sistemleri Lab. Dönem Sonu Sınavı

Tarih: 28 Mayıs 2014 Çarşamba **Süre:** 120 dakika

Sadece 5 soruyu cevaplayınız.

1. Aşağıda üç örneği gösterilen matematiksel ifadeler veriliyor. İfadedeki her bir terimin payı `1`, paydası ise herhangi bir pozitif tamsayı olabilmektedir.

```
1/2+1/3;        1/5+1/4+1/6;        1/3+1/8+1/5;
```

a) JavaCC dilini (`.jj` uzantılı dosya formatı) kullanarak bu çeşit ifadeler için bir ayrıştırıcıyı, nesne ağacını üreten Java yada C++ ifadelerini de ekleyerek tanımlayınız. (10p)

b) `intanceof` işlecini kullanarak bu çeşit ifadeler için bir değerlendirici yazınız. (10p)

2. **a)** Komut satırından girilen saniye kadar child süreci bekleten bir C programı yazınız (*sleep()* süreç bekletme fonksiyonunu kullanabilirsiniz) (10p)

b) `ls > LS` komutunun yapabileceği sistem çağrılarını sırasıyla yazınız. Yapılan toplam sistem çağrısı sayısı belirlenebilir mi? Belirlenebilirse kaç adet sistem çağrısı yapmıştır? (10p)

3. LZW kodlamada optimum BITS, dictionary tablosunun son dolduğu andaki kod boyu varsayıldığında `gcBook.txt` ve `dcBook.txt` için gerçekten de en iyi veri sıkıştırma oranları (VSO) bulunurken; `dxBook.txt` için en iyi oran 17'nin 1 fazlası 18 bitte elde edilmiştir. Başka bir deyişle 2 dosya için tablonun son dolduğu, 1 dosya için de ilk boşluk kaldığı andaki kod boyu en iyi VSO'nı verir. Nedenini açıklayınız. (20p)

Dosya Adı	Optimum BITS	BITS için VSO	BITS+1 için VSO	BITS+2 için VSO
dxBook.txt	**17**	**%40.12**	**%39.59**	**%41.79**
gcBook.txt	**18**	**%38.32**	**%40.12**	**%42.23**
dcBook.txt	**19**	**%38.97**	**%40.79**	**%42.82**

4. **a1)** Yanda verilen thread fonksiyonlarının eşzamanlı olarak icrası tamamlandığında, $X \leq Y$ olmaktadır. **Y** değerinin kaç olacağını açıklayınız. (3p)

X=0 başlangıç değeri için;

```
ThA () { for (i=0; i<5; i++)  {X = X + 1;}
ThB () { for (j=0; j<5; j++)  {X = X + 2;}
```

a2) İki thread'in işlemleri tamamlandığında, $X \neq 1$ eşitliğinin neden gerçekleşemeyeceğini belirtiniz. (3p)

a3) X değişkeni sadece {5,7,9,11,13,15} değerlerini alacak şekilde gerekli düzenlemeleri yapınız. (6p)

b) A, B, C ve D ilk değerleri sıfır olan eşzamanlı işlem parçacıkları tarafından ortak olarak kullanılan değişkenlerdir. Buna göre, yanda verilen dört fonksiyon eşzamanlı çalıştığında oluşan ekran çıktısını açıklayınız. (8p)

```
fnkA () { printf ("a");  A = 1; while (D = = 0) {} printf ("A"); }
fnkB () { while (A= = 0) {}
          printf ("b"); B=1;  while (D= = 0) {} printf ("B"); }
fnkC () { while (B= = 0) {}
          printf ("c"); C=1;  while (D= = 0) {} printf ("C"); }
fnkD () { while (C= = 0) {}  printf ("d"); D=1;  printf ("D"); }
```

5. Infix notasyonu kullanan hesap makinesi programına incOp operatörünün eklenmesi istenmektedir. incOp operatörü 2 farklı şekilde çalışmaktadır. "incOp Sayı" kullanımı Sayı+1 değerini, "Sayı1 incOp Sayı2" kullanımı ise Sayı1+Sayı2 değerini üretmektedir. Hesap makinesinin incOp operatörüne ilave olarak + ve - operatörlerini desteklediği varsayıldığında operatör öncelikleri yüksek öncelikten düşük önceliğe doğru sırasıyla incOp (tekli operand durumu), incOp (ikili operand durumu), '+', '-' şeklinde değişmesi gerekmektedir. incOP, '+' ve '-' operatörlerine karşılık düşen öncelik tanımlamalarını ve gramer kurallarını belirterek karşılığı olan semantik aksiyonları yazınız. (20p)

Not: Operatör birleşme özellikleri (associativity) istenildiği şekilde seçilebilir. Gramer kurallarındaki aritmetik ifadeleri temsil etmek için *exp* terminal-olmayan sembolü kullanılabilir.

6. Paylaşımlı bellek (shared memory) yöntemine örnek bir kod parçası yanda tarafta verilmiştir. Bu kod parçasına göre aşağıdaki soruları cevaplayınız. (20p)

a) shmget, shmat ve shmdt fonksiyonlarının işlevlerini birer cümle ile açıklayınız.

b) segment büyüklüğünün byte cinsinden eşitini bulunuz.

c) Ekran çıktısını veriniz.

```
int main () {
  int segment_id;  char* shared_memory;
  struct shmid_ds shmbuffer;  int segment_size;
  const int shared_segment_size = 0x6400;
  segment_id =
      shmget (IPC_PRIVATE, shared_segment_size, . . . );
  shared_memory = (char*) shmat (segment_id, 0, 0);
  shmctl (segment_id, IPC_STAT, &shmbuffer);
  printf ("Merhaba Dunya\n");
  sprintf (shared_memory, "Hello, world.");
  shmdt (shared_memory); }
```

7. **a)** Her biri 2^n adet istemciye sahip n adet sunuculu istemci sunucu mimarisini oluşturabilmek için kaç adet bilgisayara ihtiyaç vardır? Açıklayınız. (10p)

b) Sunucu ve İstemci bilgisayar üzerinde çalışacak olan programların yapması gereken adımları akış diyagramında sistem çağrılarıyla birlikte yazınız. (10p)

2013-2014 Bilgisayar Sistemleri Laboratuarı Final Sınavı Cevapları

***** Cevap 1 *****

a)

Gramer kuralları

```
E -> TE'
E' -> "+" TE'
E' ->
T -> "1" "/" num
```

JavaCC tanımlaması

Art.jj

```
options {
    STATIC = true; // make parser
methods static
//    DEBUG_PARSER = true;
    LOOKAHEAD = 1;
}

PARSER_BEGIN(Func)
public class Func {
}
PARSER_END(Func)

SKIP : { " " | "\t" | "\r" }

TOKEN : /* OPERATORS */
{
    < #DIGIT: ["0"-"9"] >
  | < ONE: "1" >
  | < NUM: (<DIGIT>)+ >
  | < PLUS: "+" >
  | < DIV: "/" >
  | < EOL : "\n" >
}

Exp parse() :
{ Exp a; }
{ a=E()(<EOF> | <EOL>) { return a; }
}

Exp E() :
{ Exp a, b; }
{ a=T() (
   <PLUS>  b=T() { a=new Plus(a, b); }
  )*
  { return a; }
}

Exp T() :
{ Token t; }
{ <ONE> <DIV> t=<NUM>
  { return (new
Num(Integer.parseInt(t.image))); }
}
```

AST.java

```
class Exp {  }

class Plus extends Exp{
  Exp a, b;
  public Plus(Exp x, Exp y) {
    a = x;
    b = y;
  }
}

class Num extends Exp {
  int n;
  public Num(int x) {
      n = x;
  }
}
```

EvalArt.java

```
public class EvalArt {
 static int pay, payda;
 public  static  void  main(String[]
   args) {
  Art parser = new Art(System.in);
  try {
   EvalArt art = new EvalArt();
   art.eval(parser.parse());
   System.out.println(pay  +  "/"  +
   payda);
  } catch(ParseException e) {
     e.printStackTrace();
  }
 }

 public void eval(Exp exp) {
  if (exp instanceof Plus) {
   eval(((Plus)exp).a);
   int a_pay = pay;
   int a_payda = payda;
   eval(((Plus)exp).b);
   int b_pay = pay;
   int b_payda = payda;
   pay =
     a_pay*b_payda+ b_pay*a_payda;
   payda = a_payda*b_payda;
  }
  else { //if (exp instanceof Num)
   pay = 1;
   payda = ((Num)exp).n;
  }
 }
}
```

***** Cevap 2 *****

a)

```
int main (int argc, char **argv) {
 int child_id, seconds;
 if (argc == 1)
```

b)

Bir komutun yaptığı sistem çağrılarının sayısı, onun kaynak kodundaki görünür ifadelerle sınırlı değildir. Kütüphane dosyalarının açılıp okunması ve komutu

```
  seconds = DEFAULT_TIME;
 else
  seconds = atoi (argv[1]);
 printf ("Here I am in the program!  Seconds = %d\n",
seconds);
 child_id = fork();
 if (child_id) {
  printf ("I'm the parent.  Bye now!\n");
 }
 else {
  sleep(seconds);
  printf ("I'm the child.  Bye now!\n");
 }
 return 0;
}
```

koşacak sürecin yönetimi gibi gereksinmeleri karşılamak için gerekli bazı kod parçaları da komutun derlenmesi esnasında icra edilebilir koda dahil edilir. Bu kod parçaları birçok sistem çağrısının yapılmasına ihtiyaç duyar.

******* Cevap 3 *******

Dosya dictionary tablosunun son dolduğu BITS değeri ile kodlanırken, dosyada sonlara yaklaşılmış ama tamamı kodlanamadan tablo dolmuştur. Dolayısıyla dosyanın geri kalan kısmından tabloya yeni kodlar eklemek mümkün olamamıştır. Yeni kodlar eklendikçe 1 karakter eksiği çıkışa yazılacak ve onun karakter boyu oranında veri sıkıştırma oranı iyileşecekti. Öte yandan tabloda ilk boşluk kaldığı BITS değerinde yeni kodların tamamı eklenmesine rağmen herbir veri çıkışa tablonun son dolduğu BITS değerinden 1 bit fazla yazılacaktır.

`gcBook.txt` ve `dcBook.txt` dosyalarında kod boyu arttırmanın her bir kodu çıkış dosyasına 1 bit fazla yazma dezavantajı, tabloya yeni kodların tamamının eklenebilmesi avantajına baskın çıkmıştır. `dxBook.txt` için ise tersi olmuştur. Yani yeni kodları getirdiği avantaj baskın çıkmıştır. Dosyanın bu BITS değerlerinden hangisi için en iyi veri sıkıştırma oranını vereceği kodlamadan belirlenemez. O yüzden bazı dosyalarda son dolduğu bazılarında da ilk boşluk kaldığı BITS değeri ile kodlamak uygundur.

******* Cevap 4 *******

a) `fork()` => yeni çocuk süreç oluşur. Çocuk süreç farklı bir bellek alanı kullanır. İstenildiği zaman `kill` komutuyla durdurulabilir.

b) socket(), bind(), listen(), accept(), read() ve close()
socket(), connect(), send(), recv(), close()

******* Cevap 5 *******

```
//Öncelikler
left '-'
left '+'
left 'incOP'  /*ikili operand durumu*/
left INCOP /*tekli operand durumu*/
```

```
//Gramer kuralı
exp: exp 'incOP' exp  {$$ = $1+$3;}  /*ikili operand
durumu için gramer kuralı*/
exp: 'incOP' exp %prec INCOP {$$ = $2+1}  /*tekli
operand durumu için gramer kuralı*/
```

******* Cevap 6 *******

a) **shmget** : Paylaşımlı bellek bölgesini tahsis eder.
shmat : Paylaşımlı bellek bölgesi ile süreç adres uzayını, sürecin kullanımı için ilişkilendirir.
shmdt : Ayrılan paylaşımlı bellek segmentini serbest bırakır.
b) 25600 byte
c) Merhaba Dunya

******* Cevap 7 *******

```
Arac() {
 for(int i=0;i<Y;i++) {
 artır(yolcu_sırası);  //en az X tane
yolcu sırada bulunmalı
 azalt(yolcu_bindir); //başlangıç değeri X
 servis();
 for(int i=0;i<X;i++) {
  artır(arac_servis); //yolcu indirilmesi
  azalt(yolcu_indir); //her bir yolcunun
inişinin tamamlanması beklenecektir.
}
}
}
```

```
Semafor  yolcu_sırası
Semafor  yolcu_bindir
Semafor  arac_servis
Semafor  yolcu_indir

Yolcu() {
 while(true) {
  azalt(yolcu_sırası);
  artır(yolcu_bindir);
  servis2();
  azalt(arac_servis);
  artır(yolcu_indir);
 }
}
```

Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
2014-2015 Bahar Yarıyılı
Bilgisayar Sistemleri Lab. Dönem Sonu Sınavı

Tarih: 28 Mayıs 2015 Perşembe **Süre:** 120 dakika

Sadece 5 soruyu cevaplayınız.

1. Tamsayılar ve x değişkeni üzerinde tanımlı toplama ve faktöriyel alma işlemlerinden oluşan matematiksel ifadeleri üretebilen bir gramerin kuralları ile sözdizim sınıfları aşağıdaki tabloda veriliyor.

Gramer Kuralları	Sözdizim Sınıfları
`E –> E "+" E`	`(Plus)`
`E –> " (" E ")"`	`(Par)`
`E –> "(" E ")" "!"`	`(Fact)`
`E –> "x"`	`(Var)`
`E –> ["0" - "9" ]+`	`(Num)`

Yukarıdaki gramer kuralları ve sözdizim sınıflarına göre aşağıdaki soruları cevaplayınız.

a) Bu gramer aşağıda verilen üç ifadeden hangilerini üretebilir? **(5p)**
`(x + 21) + (x + 2)!` `x + ((x + 17)! + x + 2)!` `x! + ((13 + x)! + (29)!)!`

b) Grameri LL(1) gramerine dönüştürünüz. **(5p)**

c) `Par` ve `Fact` kurallarını JavaCC'de tanımlayınız**. (5p)**

d) Aşağıdaki ifade için soyut sözdizim ağacını nesne biçiminde oluşturunuz. **(5p)**
`x + (5 + (x + 2)!)!`

2. **a)** `/proc` dosya sisteminde bulunan `premptyset` ve `praddset` fonksiyonlarının işlevlerini birer cümle ile açıklayınız. **(10p)**

b) Putty programı ile T1, T2 ve T3 olmak üzere üç terminal açıldığı varsayılarak,
b1) "truss truss" komutunun çıktısı T2 terminali içinde görünürken, Unix komutunun çıktısı T3 terminaline yönlendirilebilmesi için gereken komut dizisini yazınız (T2 ve T3 terminallerinde yazılan komutları ayrı ayrı belirtiniz). **(5p)**

b2) T1 terminalinden T2 terminali gözetlenmek (çocuk süreçleri de gözetleyecek şekilde) isteniyor. Bunun için gerekli komut dizisini oluşturunuz. .(T1 ve T2 terminallerinde yazılan komutları ayrı ayrı belirtiniz) **(5p)**

3. Linux makine üzerinde gcc komutu hangi amaçla kullanılmaktadır? Aşağıda gösterilen ve gcc komutunun parametre olarak aldığı komut satırı argümanları ne anlama gelmektedir? **(20p)**

```
linux-1yy1:~> gcc -o sample sample.c
```

`komut satırı argümanları`

4. "Prefix" notasyonda `'+'`, `'-'`, `'*'`, `'/'` işlemleri için gramer kurallarını belirterek karşılığı olan semantik aksiyonları yazınız. **(10p)**

Bu yazdığınız grammer çalıştırılıp konsoldan aşağıdaki ifadeler girildiğinde her biri için çıktılar (varsa hata mesajları) ne olur? **(10p)**

```
+ - 1 2 1 = ?
- + 1 2 1 = ?
```

Not: Operatör birleşme özellikleri (*associativity*) istenildiği şekilde seçilebilir. Gramer kurallarıyla aritmetik ifadeleri temsil etmek için `exp` terminal-olmayan sembolü kullanılabilir.

5. Yeşil ve kırmızı rengi içeren ***tek bir trafik ışığı*** sistemi tasarlanacaktır. Trafik ışığı bir süre kırmızı yandıktan sonra bir süre de yeşil yanacaktır. Trafik ışığının durum döngüsü sürekli olarak bu şekilde gerçekleşirken, yeşil yandığında bekleyen ve yeni gelen arabalar geçiş yapabilecektir. Araç geçişi varken trafik ışığı kırmızı yanmayacaktır (kırmızı ışık yandığında da araç geçişi duracaktır). Bu sistemi modelleyen fonksiyonları sadece bir tane semafor senkronizasyon nesnesi kullanarak yazınız. **(20p)**

6. LZW kodlamada en iyi veri sıkıştırma oranını veren optimum BITS (kodboyu) değeri neden bazı dosyalar için dictionary tablosunun son dolduğu; bazıları için de tabloda ilk boşluk kaldığı andaki kod boyudur? Açıklayınız. **(20p)**

Diğer soru arka sayfadadır →

7. Aşağıdaki kod blokları, süreçler arası haberleşme tekniklerinden biri olan Haritalanmış Bellek yöntemi ile yazılmıştır. Gerekli kütüphanelerin tanımlandığını varsayınız. Aşağıda kaynak kodu gösterilen programlar arasında önce `mmap-write.c` ve sonra `mmap-read.c` koşmaktadır.

mmap-write.c

```
int main (intargc, char* constargv[]){

 int fd;
 void* file_memory;
 fd = open (argv[1], O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);
 file_memory = mmap (0, FILE_LENGTH, PROT_WRITE, MAP_SHARED, fd,0);
 close (fd);
 char string1[25600]=" SON  ";
 sprintf((char*) file_memory, "%s\n",string1);
 sprintf (file_memory,"%d\n", 150 );
 printf("%s\n"," Sistem Lab. ");
 munmap (file_memory, FILE_LENGTH);
 return 0;
}
```

mmap-read.c

```
int main (intargc, char* constargv[]){

 int fd;
 void* file_memory;
 int integer;
 fd = open (argv[1], O_RDWR, S_IRUSR | S_IWUSR);
 file_memory= mmap (0, FILE_LENGTH, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
 close (fd);
 sscanf (file_memory,"%d",&integer);
 printf ("dosyaicerigi: %d\n", integer*2);
 printf ("%s\n ", " :) ");
 return 0;
}
```

a) `mmap-write.c` programı çalıştığında oluşacak ekran çıktısını veriniz. **(5p)**

b) `mmap-write.c` programı çalıştığında oluşacak dosya içeriğini veriniz. **(5p)**

c) `mmap-read.c` programı çalıştığında oluşacak ekran çıktısını veriniz. **(5p)**

d) `mmap-read.c` programı çalıştığında oluşacak dosya içeriğini veriniz. **(5p)**

2014-2015 Bilgisayar Sistemleri Laboratuarı Final Sınavı Cevapları

***** Cevap 1 *****

a)
Birinci ifade üretilebilir.
İkinci ifade üretilebilir.
Üçüncü ifade üretilemez. (x! teriminden dolayı)

b)
```
E  –> P E'
E' –> + P E'  |
P  –> ( E ) P' | x
P' –> ! |
```

c)
```
void E() {}
{ P() ( "+"P())* }

void P() {}
{ "x" | <NUM> | "(" E() ")" ("!")? }
```

d)
```
Exp exp =
  new Plus(new Var(),
   new Fact(new Plus(
    new Num(5),
     new Fact(new Plus(
      new Var(), new Num(2))))));
```

***** Cevap 2 *****

a)
premptyset(&set); /* turn off all flags in set */
praddset(&set, flag); /* turn on the specified flag */

b1) T3: tty
T2 : truss truss > T3 terminali ismi
b2) T2: echo $$
T1: truss –o dosya –f –p T2PID

***** Cevap 3 *****

gcc komutu Linux makine üzerinde birden fazla .c uzantılı kaynak dosyanın tek bir çalıştırabilir dosya haline dönüştürmek için kullanılan derleme komutudur.
sample → c uzantılı dosyaların derlenerek elde edilmiş çalıştırılabilir dosya
sample.c → c uzantılı kaynak dosya

***** Cevap 4 *****

```
exp: NUM          { $$ = $1;   }
   | '+' exp exp { $$ = $2 + $3; }
   | '-' exp exp { $$ = $2 - $3; }
   | '*' exp exp { $$ = $2 * $3; }
   | '/' exp exp { $$ = $2 / $3; }
   ;
```

Çıktılar:
```
+ - 1 2 1 = 0
- + 1 2 1 = 2
```

***** Cevap 5 *****

```
trafik-isigi () {
  ...
  WaitForSingleObject(trafik_isik_mutex, INFINITE);
  printf( "Kirmizi...\n\n " ) ;
  isik_rengi = KIRMIZI ;
  Sleep( KIRMIZI_SURE ) ;
  printf( "YESIL\n\n " ) ;
  ReleaseMutex(trafik_isik_mutex);
  isik_rengi = YESIL ;
  Sleep( YESIL_SURE ) ; }
  ...
}

araclar() {
  ...
  WaitForSingleObject(trafik_isik_mutex, INFINITE);
  printf( "Araba geciyor %d...\n\n",GetCurrentThreadId() ) ;
  Sleep( ARABA_SURE ) ;
  printf( "Araba gecti %d...\n\n",GetCurrentThreadId() ) ;
  ReleaseMutex(trafik_isik_mutex);
  ...
}

HANDLE trafik_isik_mutex;
...
main() {
```

```
  ...
  DWORD ThreadID,ThreadID2;
  trafik_isik_mutex = CreateMutex(NULL,FALSE,NULL);
  CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE) trafik-isigi,NULL, 0, &ThreadID);

  printf("isiklarbaslatildiid= %d\n",ThreadID);

  for( i = 0 ; i<arabasayisi ; i++ ) {
    CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)araclar ,NULL, 0, &ThreadID2);
    printf("Araba geldi  id= %d\n",ThreadID2);
  }
...
}
```

******* Cevap 6 *******

Dosya, dictionary tablosunun son dolduğu BITS değeri ile kodlanırken, dosyada sonlara yaklaşılmış ama tamamı kodlanamadan tablo dolmuştur. Dolayısıyla dosyanın geri kalan kısmından tabloya yeni kodlar eklemek mümkün olamamıştır. Yeni kodlar eklendikçe 1 karakter eksiği çıkışa yazılacak ve onun karakter boyu oranında veri sıkıştırma oranı iyileşecekti. Öte yandan tabloda ilk boşluk kaldığı BITS değerinde yeni kodların tamamı eklenmesine rağmen herbir veri çıkışa tablonun son dolduğu BITS değerinden 1 bit fazla yazılacaktır. Bazı dosyalarda kod boyu arttırmanın her bir kodu çıkış dosyasına 1 bit fazla yazma dezavantajı, tabloya yeni kodların tamamının eklenebilmesi avantajına baskın çıktığından bazılarında da tersi söz konusu olduğundan optimumu kod boyu sırasıyla tablonun son dolduğu veya ilk boş kaldığı andaki kod boyu olmaktadır.

******* Cevap 7 *******

```
a) Sistem Lab.
b) 150
c) dosya icerigi: 300
   :)
d) 150
```