Instructions:
1. Print your name in the space provided
2. Print your student identifier in the space provided.
3. Print the section number of the section in which you are enrolled in the space provided.
4. Print the date in the space provided.
5. You have one hour and fifty minutes (110 minutes) to complete this examination.
6. You may use a calculator.
7. Reference materials are prohibited. You must complete this test without the aid of course notes or reference texts.
8. Questions requiring written answers must be answered using standard American English. Answers containing spelling or grammatical errors will be given no credit.
9. Answers must be coded legibly. Answers that cannot be read by your instructor will be given no credit.
10. You must do your own work.

Scoring:
1. The table to the right lists the number of raw points available for each problem and the total number of raw points that can be earned on this test.
2. Your score will be normalized to a fraction of 350 points. If \( n \) is your normalized score, \( r \) is your raw score, and \( T \) is the total number of raw points, then \( n = \frac{350r}{T} \).
1. (30 points) Write function `isupper` that returns `1` (true) if the input integer parameter `c` has an integer value equal to one of the ASCII character codes for a capital letter.

```c
int isupper(int c)
{
    return 'A'<=c&&c<='Z';
}
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>return type int</td>
<td>5</td>
</tr>
<tr>
<td>parameter type int</td>
<td>5</td>
</tr>
<tr>
<td>parameter name c</td>
<td>5</td>
</tr>
<tr>
<td>function name isupper</td>
<td>5</td>
</tr>
<tr>
<td>correct return value</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>30</strong></td>
</tr>
</tbody>
</table>

2. (60 points) Write function `decval` that returns an integer equivalent to the decimal value of a digit in base 36. Digits in base 36 are represented as a single character. Digits in base 36 consist of the decimal digits and the alphabetic letters. You may assume the existence of functions `isupper`, `islower`, and `isdigit`. Prototypes for functions `islower` and `isdigit` are given below. Function `isupper` returns `1` (true) if the input integer parameter `c` has an integer value equal to one of the ASCII character codes for a capital letter. Function `islower` is similar to `isupper`. Function `islower` returns `1` (true) if the input integer parameter `c` has an integer value equal to one of the ASCII character codes for a lower case letter. Function `isdigit` returns a `1` (true) if the input integer parameter `c` has an integer value equal to one of the ASCII character codes for a digit.

```c
int isupper(int c)
int isdigit(int c)

int decval(int c)
{
    if (islower(c)) return c-'a'+10;
    if (isupper(c)) return c-'A'+10;
    if (isdigit(c)) return c-'0';
}
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>return type int</td>
<td>5</td>
</tr>
<tr>
<td>parameter type int</td>
<td>5</td>
</tr>
<tr>
<td>parameter name c</td>
<td>5</td>
</tr>
<tr>
<td>function name decval</td>
<td>5</td>
</tr>
<tr>
<td>if (islower(c))</td>
<td>5</td>
</tr>
<tr>
<td>return c-'a'+10</td>
<td>10</td>
</tr>
<tr>
<td>if (isupper(c))</td>
<td>5</td>
</tr>
<tr>
<td>return c-'A'+10</td>
<td>10</td>
</tr>
<tr>
<td>if (isdigit(c))</td>
<td>5</td>
</tr>
<tr>
<td>return c-'0'</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>60</strong></td>
</tr>
</tbody>
</table>
3. (85 points) Write function \textit{decimal}. Function \textit{decimal} converts an integer value in a foreign base to decimal. Function \textit{decimal} has two parameters. The first parameter \textit{n} is a pointer to a character. Parameter \textit{n} points to the first character of a string that represents a number in a foreign base. Each character of the string is a digit in the foreign base. Letters represent values greater than nine. The second parameter \textit{b} is the value of the foreign base. Function \textit{decimal} returns an integer equivalent to the decimal value of number \textit{n} in foreign base \textit{b}. For example, program p01 in Figure 1 prints 11181.

You may use any of the functions discussed or implemented in problems 1 and 2.

Include all necessary include files.

```c
#include <stdio.h>

int main()
{
    printf("%d\n", decimal("2bad",16));
    return 0;
}
```

\textbf{Figure 1.} Program p01.

```
#include <stdio.h>

t
```
4. (60 points) What is printed in file o04.dat by program p04 listed below? Please put your answer on the following page

```c
#include <stdio.h>
typedef enum day_t
    {first, second, third, fourth, fifth, sixth, seventh,
     eighth, nineth, tenth, eleventh, twelfth}
day_t;
static char *day[]= 
    {"first", "second", "third", "fourth", "fifth", "sixth", "seventh",
     "eighth", "nineth", "tenth", "eleventh", "twelfth"};
static char *gift[]= 
    {"a parrot in a monkeepuzzle tree",
     "two toothsome tigers",
     "three fertile ferrets",
     "four angry alligators",
     "five fluorescent flamingos",
     "six sickly sows",
     "seven chattering chimpanzees",
     "eight turkey vultures devouring",
     "nine frogs croaking",
     "ten cockatoos molting",
     "eleven bees a'buzzing",
     "twelve blizzards blowing"};
day_t skipday(day_t d) { return (d+7)%12; }
int main() 
{ FILE *o=fopen("o04.dat","w");
  day_t d; int c;
  for (c=0,d=first;c<6;c++,d=skipday(d)) 
    { fprintf(o,"On the %s",day[d]);
      fprintf(o," day of Christmas my true love gave to me\n");
      fprintf(o,"%s\n",gift[d]);
    }
  return 0;
}
```

Figure 2. Program p04
5. (50 points) Solve the equation given below for \( R \).

\[
P = \frac{R}{(1+i)} + \frac{R}{(1+i)^2} + \cdots + \frac{R}{(1+i)^n}
\]

\( (1) \) \( P = R(1+i)^{-1} + R(1+i)^{-2} + \cdots + R(1+i)^{-n} \)

\( (2) \) \( (1+i)P = R + R(1+i)^{-1} + \cdots + R(1+i)^{-n+1} \)

\( (3) \) \( (1+i)P - P = R - R(1+i)^{-n} \)

\( (4) \) \( Pi = R[1 - (1+i)^{-n}] \)

\( (5) \) \( R = \frac{P i}{1 - (1+i)^{-n}} \)
6. (30 points) Convert $1,012,010_{10}$ to base 36. Show all your work.

<table>
<thead>
<tr>
<th>divisor</th>
<th>dividend</th>
<th>quotient</th>
<th>remainder</th>
<th>digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>1012010</td>
<td>28111</td>
<td>14</td>
<td>e</td>
</tr>
<tr>
<td>36</td>
<td>28111</td>
<td>780</td>
<td>31</td>
<td>v</td>
</tr>
<tr>
<td>36</td>
<td>780</td>
<td>21</td>
<td>24</td>
<td>o</td>
</tr>
<tr>
<td>36</td>
<td>21</td>
<td>0</td>
<td>21</td>
<td>l</td>
</tr>
</tbody>
</table>

$1,012,010_{10} = \text{love}_{36}$

7. (30 points) Compute the monthly payment for person borrowing $100,000 for 30 years at 5 APR.

$$ R = P \frac{i}{1 - (1 + i)^{-n}} \quad i = \text{APR} / 1200, n = 12 \times \text{years}, P = \$100,000 $$

$$ R = \$536.82 $$
Problems 8, 9, 10, and 11 define portions of a queue. A queue is similar to a line of customers waiting to be served. Customers wait at the end of the line. The newest customer is placed at the end of the line. The oldest customer is at the front of the line. The oldest customer is served first. A queue is often called a FIFO data structure. FIFO is an acronym that stands for First In First Out.

8. (30 points) Code declarations for the anchor of a queue. Declarations for the anchor of a queue appear in file `queue.h`. The anchor is diagrammed in Figure 3.

```
typedef struct queue_t {
    struct queue_e *oldest;
    struct queue_e *newest;
} queue_t;
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef struct queue_t</td>
<td>5</td>
</tr>
<tr>
<td>struct queue_e *oldest</td>
<td>10</td>
</tr>
<tr>
<td>struct queue_e *newest</td>
<td>10</td>
</tr>
<tr>
<td>queue_t</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
</tr>
</tbody>
</table>

9. (25 points) Code declarations for an element of a queue. Declarations for the anchor of a queue appear in file `queue.h`. Elements on the queue are dynamically allocated. Elements are linked from oldest to newest. An element is diagrammed in Figure 4.

```
typedef struct queue_e {
    struct queue_e *newer;
    int v;
} queue_e;
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef struct queue_e</td>
<td>5</td>
</tr>
<tr>
<td>struct queue_e *newer</td>
<td>10</td>
</tr>
<tr>
<td>int v</td>
<td>5</td>
</tr>
<tr>
<td>queue_t</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>25</td>
</tr>
</tbody>
</table>
10. (45 points) Code function `queuecreate`. Function `queuecreate` is called in program `p09` shown in Figure 5. Figure 6 depicts the actions of function `queuecreate` showing a diagram of data before and after the function is called.

```c
#include <queue.h>

int main()
{
    queue_t *q=queuecreate();
    return 0;
}
```

![Figure 5. Program p09.](image)

![Figure 6. Actions of function `queuecreate`](image)

```c
queue_t *queuecreate(void)
{
    queue_t *q=(queue_t*)malloc(sizeof(queue_t));
    q->oldest=NULL;
    q->newest=NULL;
    return q;
}
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>queue_t *q</td>
<td>5</td>
</tr>
<tr>
<td>(queue_t*)</td>
<td>5</td>
</tr>
<tr>
<td>malloc(sizeof(queue_t))</td>
<td>10</td>
</tr>
<tr>
<td>q-&gt;oldest=NULL</td>
<td>10</td>
</tr>
<tr>
<td>q-&gt;newest=NULL</td>
<td>10</td>
</tr>
<tr>
<td>return q</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>45</strong></td>
</tr>
</tbody>
</table>
11. (95 points) Code function `queuedestroy`. Function `queuedestroy` is called in program `p10` shown in Figure 7. Figure 8 depicts the actions of function `queuedestroy` showing a diagram of data before and after the queue is destroyed. Elements on the queue are dynamically allocated. Elements are linked from oldest to newest. Put your answer on the following page.

```c
#include <queue.h>
int main()
{
    queue_t *q=queuecreate();
    queueinsert(q,5);
    queueinsert(q,10);
    queuedestroy(q);
    return 0;
}
```

**Figure 7.** Program `p10`.

```
#include <queue.h>
int main()
{
    queue_t *q=queuecreate();
    queueinsert(q,5);
    queueinsert(q,10);
    queuedestroy(q);
    return 0;
}
```

**Figure 8.** Queue before and after the queue is destroyed
Problem 11 continued. Put your answer here.

```c
static void queuekill(queue_e *e)
{
    while (e) {
        queue_e *p=e;
        e=e->newer;
        free (p);
    }
}

void queuedestroy(queue_t *q)
{
    queuekill(q->oldest);
    free(q);
}
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void queuekill</td>
<td>15</td>
</tr>
<tr>
<td>1st parameter queue_e</td>
<td>10</td>
</tr>
<tr>
<td>while (e)</td>
<td>5</td>
</tr>
<tr>
<td>queue_e *e=p;</td>
<td>10</td>
</tr>
<tr>
<td>e=e-&gt;newer;</td>
<td>10</td>
</tr>
<tr>
<td>free (p)</td>
<td>5</td>
</tr>
<tr>
<td>void queuedestroy</td>
<td>10</td>
</tr>
<tr>
<td>1st parameter queue_t</td>
<td>10</td>
</tr>
<tr>
<td>queuekill</td>
<td>5</td>
</tr>
<tr>
<td>q-&gt;oldest</td>
<td>10</td>
</tr>
<tr>
<td>free(q)</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>95</strong></td>
</tr>
</tbody>
</table>