(i) (4 pts.)

Consider the following C program.

```c
#include <stdio.h>

#define R 3
#define C 4
#define START 1000

int mat[R][C] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

int main(void) {
    int x = 0, y = 0, z = 0;

    for (x = 0; x < R; ++x) {
        z = START;
        for (y = 0; y < C; ++y) {
            if (mat[x][y] < z) {
                z = mat[x][y];
            }
        }
        printf("%d ", z);
    }
    printf("\n");
    return 0;
}
```

Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem only on this page and (if more space is needed) the next page.
This page is reserved as extra space for working on and writing your solution to Question (i).
(ii) (5 pts.)

Consider the following C program.

```c
#include <stdio.h>
#include <string.h>

#define SIZE 20

int main(void) {
    char s1[] = "xya$%fsp\my";
    char s2[] = "\m@psdf22nnz";
    char t1[SIZE];
    char t2[SIZE];
    int i = 0;

    for (i = 0; i < strlen(s1); i += 2) {
        t1[i / 2] = s1[i];
        t2[i / 2] = s2[i];
    }
    t1[i / 2] = '\0'; /* null character */
    t2[i / 2] = '\0'; /* null character */

    printf(t1, s2);
    putchar('
');
    printf(t2, s1);
    putchar('
');
    printf(t1, t1);

    return 0;
}
```

Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem only on this page and (if more space is needed) the next page.
This page is reserved as extra space for working on and writing your solution to Question (ii).
(iii)  (6 pts.)

Consider the following C program, which includes three function definitions, including the main function.

```c
#include <stdio.h>
#include <stdlib.h>

struct data {
    int v;
    struct data *next;
};

void setup(struct data **h, int *a, int len) {
    struct data *p = NULL;
    struct data *prev = NULL;
    struct data *start = NULL;
    int i = 0;
    for (i = 0; i < len; i++) {
        p = malloc(sizeof(struct data));
        p->v = a[i];
        if (prev == NULL) {
            start = p;
        } else {
            prev->next = p;
        }
        prev = p;
    }
    p->next = start;
    (*h) = start;
}

int f(struct data *h, int limit) {
    if (limit <= 0) {
        return h->v;
    } else {
        limit -= h->v;
        printf("%d ", h->v);
        return h->v + f(h->next, limit);
    }
}

int main(void) {
    int values[] = {4, 3, 8, 7, 5, 6, 2};
    const int length1 = 7, length2 = 4, limit1 = 10, limit2 = 25;
    struct data *h = NULL;
    int result = 0;
    setup(&h, values, length1);
    result = f(h, limit1);
    printf("\nresult = %d\n", result);
    setup(&h, values, length2);
    result = f(h, limit2);
    printf("\nresult = %d\n", result);
    return 1;
}
```
Show the complete output as it appears on standard output. Show all work, and clearly indicate your solution. Show your work and your solution for this problem only on this page and the previous page.
Consider the following structure definition.

```c
struct vect {
    int *elements;
    int length;
};
```

Consider also the following function prototype and its associated header comment.

```c
/* Return the inner product of two vectors with circular traversal of the vector with fewer elements (the shorter vector) in case the vectors do not have equal lengths. If either vector length is less than or equal to zero, then display an appropriate error message to standard error and exit the program. If the vector lengths are equal, then return the inner product of the two vectors. If the vector lengths are not equal, then traverse the elements of the shorter vector in a circular fashion while computing the sum of products of corresponding vector elements. For example, if v1 = (3, 2, 4, 2, 5) and v2 = (8, 1), then the function should return 3 x 8 + 2 x 1 + 4 x 8 + 2 x 1 + 5 x 8 = 100. */

int ip(struct vect *v1, struct vect *v2);
```

Develop a complete C code implementation of the function `ip`. You may define one or more additional functions that are called by `ip`.

Overflow checking in the sum of products computation is not required in your solution.

Write your solution to this problem only on this page and (if more space is needed) the next page.
This page is reserved as extra space for working on and writing your solution to Question (iv) .
Software Qualifying Exam Solutions
Spring 2015
Dept. of ECE, University of Maryland, College Park
12/10/2014

Problem 1:
1 5 9

Problem 2:
xa\m%psdf22nnz\y
\xya$s%fsp\myf2n
xaxa%s\y\y

Problem 3:
4 3 8
result = 22
4 3 8 7 4
result = 29
Problem 4:

```c
int ip_core(struct vect *small, struct vect *large) {
    int i = 0;
    int sum = 0;

    for (i = 0; i < large->length; i++) {
        sum += large->elements[i] * 
                small->elements[i % (small->length)];
    }
    return sum;
}

int ip(struct vect *v1, struct vect *v2) {
    struct vect *small = NULL;
    struct vect *large = NULL;
    int result = 0;

    if ((v1->length <= 0) || (v2->length <= 0)) {
        fprintf(stderr, "Invalid vector length\n");
        exit(1);
    }

    if (v1->length < v2->length) {
        small = v1;
        large = v2;
    } else {
        small = v2;
        large = v1;
    }

    result = ip_core(small, large);
    return result;
}
```