Name and Surname	
Student ID	

Course
1(AAAA-BARA) ☐2 (BARB – BOTS) ☐3 (BOTT – CAR) ☐4 (CAS – CORD) ☐5 (CORE – DIF) ☐6 (DIG – FIOR) ☐7 (FIOS - GIORD)☐8 (GIORE – LANE) ☐9 (LANF – MARA) ☐10 (MARB – MOH) ☐11 (MOI – PAK) ☐12 (PAL – POLH) ☐13 (POLI – ROSA) ☐14 (ROSB–SIL) ☐15 (SIM – TR)☐16 (TS –ZZ) ☐E1 (AA – LZ) ☐E2 (MA – ZZ) ☐Poli@Home☐Es. (5 crediti)☐

Theory

Question 1

Convert the following numbers: 23 **from** base 10 **to** base 2, pure binary on 8 bits 100011 **from** base 2, 2's complement on 6 bits **to** base 10 0110100 **from** base 2 **to** base 16	Result
The most significant passages to arrive the result	

Question 2

Given the following two Boolean functions *f* and *g*, determine whether they are equivalent by constructing the truth tables: $f = \overline{(\bar{x} + \bar{y})z}$ $g = x\,y + \bar{z}$	Result
The most significant passages to arrive at the result	

Question 3

Briefly describe the internal structure of a CPU.

Programming

A text file contains the information relating to the presences in the Italian football main league (serie A). Write a C program to extract statistics from a sample of filtered data according to specific parameters.

The name of the file that contains the information on the matches is passed in the first argument of the command line. Each row of the file corresponds to a specific football player with the format as follows:

```
<SURNAME><NAME><#PRESENCES><MEDIA-VOTE><TEAM>
```

where SURNAME and NAME are two distinct fields where each contains no more than 40 characters (without spaces), #PRESENCES is the number of times that the player is present in an official match, MEDIA-VOTE is the average rating assigned to the player during his attendance, TEAM (with the length no more than 25 characters) is the name of the corresponding team that the player currently plays in. We also make the following assumptions:

- All the fields are separated from each other by one single space
- The number of the players is unknown
- There are no homonyms
- The content of the file is correct with respect to the format

The rules for filtering the data are contained in a second file with the name as *filter.txt*. In this file, each line lists the player names that must be excluded:

```
<SURNAME><NAME>
```

The maximum number of rules present in this file is 100.
According to the filter, the program must print out:

- The list of players (with the related information) whose name is **not** contained in *filter.txt*
- The player with the lowest number of presences (among the printed names)
- The player with the largest number of presences (among the printed names)

In case of ambiguity, print the first player that is found.

Example:

serieA.txt

```
Chiellini Giorgio 24 5.32 JUVE
Consigli Andrea 35 6.13 ATALANTA
Marek Hamsik 38 6.45 NAPOLI
Diamanti Alessandro 34 6.88 BOLOGNA
Vucinic Mirko 31 4.3 JUVENTUS
Basta Dusan 28 6.25 UDINESE
Perrotta Simone 14 5.7 ROMA
Giovinco Sebastiano 21 4.5 JUVENTUS
```

filter.txt

```
Beckenbauer Franz
Vucinic Mirko
Perrotta Simone
Giovinco Sebastiano
Chiellini Giorgio
```

```
C:\>prog.exe serieA.txt
Consigli Andrea 35 6.13 ATALANTA
Marek Hamsik 38 6.45 NAPOLI
Diamanti Alessandro 34 6.88 BOLOGNA
Basta Dusan 28 6.25 UDINESE

Presence min.: 28 Basta Dusan
Presence max.: 38 Marek Hamsik
```

#include <stdio.h>

FILE *fopen(char *filename, char * mode) – Opening a file (mode: “r” reading – “w” writing – “a” append)
FILE *freopen(char *filename, char * mode, FILE *file_pointer) – Reassign a file pointer to a different file
int fclose(FILE *file_pointer) – Closing a file
int feof(FILE *file_pointer) – Checks if end-of-file has been reached.
int fflush(FILE *file_pointer) - Empties file’s buffer.
int getchar(void) – Reads a character from “stdin" (keyboard)
int fgetc(FILE *file_pointer) – Gets a character from a file
char *gets(char *buffer) - Reads a line from “stdin" (keyboard)
char *fgets(char *string, int maxchar, FILE *file_pointer) - Reads a line from a file
int printf(char *format _string, …) – Writes formatted output on "stdout" (screen)
int fprintf(FILE *file_pointer, char *format_string, …) – Writes formatted output on a file.
int sprintf(char *string, char *format_string, …) – Writes formatted output on a string.
int fputc(int c, FILE *file_pointer) – Writes a character on a file.
int putchar(int c) – Writes a character on “stdout” (screen).
int puts(char *string) - Writes a string on “stdout” (screen).
int fputs(char *string, FILE *file_pointer) - Writes a string on a file.
int scanf(char *format_string, args) – Reads formatted input from “stdin” (keyboard)
int fscanf(FILE *file_pointer, char *format string, args) – Reads formatted input from a file.
int sscanf(char *buffer, char *format_string, args) – Reads formatted input from a string.
EOF – end of file (costant with negative value)
NULL - null pointer (value 0)

#include <stdlib.h>

double atof(char *string) – Converts a string into a floating point value.
int atoi(char *string) – Converts a string into an integer value.
int atol(char *string) – Converts a string into a long integer value.
void exit(int val) – Terminates the program returning the value ‘val’.
EXIT_FAILURE – constant highlighting the unsuccessful termination of the program with exit(); non zero value.
EXIT_SUCCESS - constant highlighting the successful termination of the program with exit(); zero value.

#include <string.h>

char *strcpy(char *s1, char *s2) - Copies s2 in s1. Returns s1
char *strncpy(char *s1, char *s2, size_t n) – Copies the first "n" characters of s2 in s1. Returns s1.
int strcmp(char *s1, char *s2) - Compares s1 and s2 to determine the alphabetical order (<0, s1 precedes s2, 0 equal, >0 s1 follows s2)
int strncmp(char *s1, char *s2, size_t n) – Compares the first "n" characters of two strings.
int strlen(char *string) – Determines the length of a string.
char *strcat(char *s1, char *s2, size_t n) - Links s2 to s1. Returns s1
char *strncat(char *s1, char *s2, size_t n) - Links "n" characters of s2 to s1. Returns s1
char *strchr(char *string, int c) – Finds the first occurrence of the character ‘c’ in string; returns a pointer to the first occurrence of c in s, NULL if not present.
char *strrchr(char *string, int c) – Finds the last occurrence of the character ‘c’ in string.
char* strstr(char* s, char* t) – Returns a pointer to the first occurrence of t in s. returns NULL if not present.
char* strtok(char* s, const char* t) – Decomposes s in tokens, the characters that limit the tokens are contained in t. returns the pointer to the token (NULL if any is found). At the first call the string to be decomposed is s and the characters delimiting the various tokens in t. To operate on the same string, at following calls NULL has to be passed instead of s.

#include <ctype.h>

int isalnum(int c) – True if ‘c’ is alphanumeric.
int isalpha(int c) – True if ‘c’ is an alphabetic charater.
int iscntrl(int c) – True if ‘c’ is a control character.
int isdigit(int c) - True if ‘c’ is a decimal digit.
int islower(int c) - True if ‘c’ is lowercase.
int isprint(int c) - True if ‘c’ is a printable character.
int ispunct (int c) - True if ‘c’ is a punctuation character.
int isspace(int c) - True if ‘c’ is a space character.
int isupper(int c) - True if ‘c’ is uppercase.
tolower(int c) – Converts ‘c’ to lowercase.
int toupper(int c) – Convert ‘c’ to uppercase.

#include <math.h>

int abs (int n) – integer absolute value
long labs(long n) – long absolute value
double fabs (double x) – absolute value of x
double acos(double x) - arccosine
double asin(double x) - arcsin
double atan(double x) - artangent
double atan2(double y, double x) – arctangent of y/x.
double ceil(double x) – round up value of x.
double floor(double x) – round down value of x.
double cos(double x) – cos (x in radians)
double sin(double x) – sin (x in radians)
double tan(double x) – tan (x in radians)
double cosh(double x) – hyperbolic cosine
double sinh(double x) – hyperbolic sin
double tanh(double x) – hyperbolic tangent
double exp(double x) - e^x
double log(double x) - log(x).
double log10 (double x) – logarithm base 10
double pow (double x, double y) - x^y
int rand (void) – random integer between 0 and RND_MAX.
int random(int max_num) – random value between 0 and max_num.
void srand(unsigned seed) – initialize the sequence of random values
double sqrt(double x) – square root

#include <limits.h>

INT_MAX – Maximum value that can be represented by int variable.
INT_MIN – Minimum value that can be represented by int variable.
LONG_MAX - Maximum value that can be represented by long variable.
LONG_MIN - Minimum value that can be represented by long variable.

#include <float.h>

FLT_MAX, DBL_MAX - Maximum value that can be represented by float (or double) variable.
FLT_MIN, DBL_MIN - Minimum value that can be represented by float (or double) variable.

Computer Science – 05/07/2013 – **Duration: 2h** **Turn 2– Version. B**

Name e Surname	
Student ID	

Course
1(AAAA-BARA) ☐2 (BARB – BOTS) ☐3 (BOTT – CAR) ☐4 (CAS – CORD) ☐5 (CORE – DIF) ☐6 (DIG – FIOR) ☐7 (FIOS - GIORD)☐8 (GIORE – LANE) ☐9 (LANF – MARA) ☐10 (MARB – MOH) ☐11 (MOI – PAK) ☐12 (PAL – POLH) ☐13 (POLI – ROSA) ☐14 (ROSB–SIL) ☐15 (SIM – TR)☐16 (TS –ZZ) ☐E1 (AA – LZ) ☐E2 (MA – ZZ) ☐Poli@Home☐Es. (5 crediti)☐

Theory

Question 1

Convert the following numbers: 25 **from** base 10 **to** base 2, pure binary on 8 bits 101100 **from** base 2, 2'c complement on 6 bits **to** base 10 0101101 **from** base 2 **to** base 16	Result
The most significant passages to arrive the result	

Question 2

Given the following two Boolean functions *f* and *g*, determine whether they are equivalent by constructing the truth tables: $f = \overline{(x + \overline{y})z}$ $g = \overline{x}\, y + \overline{z}$	Result
The most significant passages to arrive the result	

Question 3

Briefly describe the internal structure of a CPU.

Programming

A text file contains the information relating to the presences in the Italian football main league (serie A). Write a C program to extract statistics from a sample of filtered data according to specific parameters.

The name of the file that contains the information on the matches is passed in the first argument of the command line. Each row of the file corresponds to a specific football player with the format as follows:

```
<SURNAME><NAME><MEDIA-VOTE><TEAM><#PRESENCES>
```

where SURNAME and NAME are two distinct fields where each contains no more than 40 characters (without spaces), MEDIA-VOTE is the average rating assigned to the player during his attendance, TEAM (with the length no more than 25 characters) is the name of the corresponding team that the player currently plays in, #PRESENCES is the number of times that the player is present in an official match. We also make the following assumptions:

- All the fields are separated from each other by one single space
- The number of the players is unknown
- One player can appears more than once in different teams.
- The content of the file is correct with respect to the above format

The rules for filtering the data are contained in a second file with the name as *filter.txt*. In this file, each line lists the player names that must be excluded:

```
<SURNAME><NAME>
```

The maximum number of rules present in this file is 100.

According to the filter, the program must print out:

- The list of players (with the related information) whose name is contained in *filter.txt*
- The player with the lowest number of presences (among the printed names)
- The player with the largest number of presences (among the printed names)

In case of ambiguity, print the first player that is found.

Example:

serieA.txt

```
Chiellini Giorgio 5.32 JUVENTUS 24
Consigli Andrea 6.13 ATALANTA 35
Marek Hamsik 6.45 NAPOLI 38
Diamanti Alessandro 6.88 BOLOGNA 34
Vucinic Mirko 4.3 JUVENTUS 31
Consigli Andrea 7.33 MILAN 54
Marek Hamsik 8.1 INTER 28
Consigli Andrea 3.2 NAPOLI 3
```

filter.txt

```
Beckenbauer Franz
Vucinic Mirko
Consigli Andrea
```

```
C:\> prog.exe serieA.txt
Consigli Andrea 6.13 ATALANTA 35
Vucinic Mirko 4.3 JUVENTUS 31
Consigli Andrea 7.33 MILAN 54
Consigli Andrea 3.2 NAPOLI 3

Presence min.: 3 Consigli Andrea
Presence max.: 54 Consigli Andrea
```

#include <stdio.h>

FILE *fopen(char *filename, char * mode) – Opening a file (mode: “r” reading – “w” writing – “a” append)
FILE *freopen(char *filename, char * mode, FILE *file_pointer) – Reassign a file pointer to a different file
int fclose(FILE *file_pointer) – Closing a file
int feof(FILE *file_pointer) – Checks if end-of-file has been reached.
int fflush(FILE *file_pointer) - Empties file’s buffer.
int getchar(void) – Reads a character from “stdin" (keyboard)
int fgetc(FILE *file_pointer) – Gets a character from a file
char *gets(char *buffer) - Reads a line from “stdin" (keyboard)
char *fgets(char *string, int maxchar, FILE *file_pointer) - Reads a line from a file
int printf(char *format _string, …) – Writes formatted output on "stdout" (screen)
int fprintf(FILE *file_pointer, char *format_string, …) – Writes formatted output on a file.
int sprintf(char *string, char *format_string, …) – Writes formatted output on a string.
int fputc(int c, FILE *file_pointer) – Writes a character on a file.
int putchar(int c) – Writes a character on “stdout” (screen).
int puts(char *string) - Writes a string on “stdout” (screen).
int fputs(char *string, FILE *file_pointer) - Writes a string on a file.
int scanf(char *format_string, args) – Reads formatted input from “stdin” (keyboard)
int fscanf(FILE *file_pointer, char *format string, args) – Reads formatted input from a file.
int sscanf(char *buffer, char *format_string, args) – Reads formatted input from a string.
EOF – end of file (costant with negative value)
NULL - null pointer (value 0)

#include <stdlib.h>

double atof(char *string) – Converts a string into a floating point value.
int atoi(char *string) – Converts a string into an integer value.
int atol(char *string) – Converts a string into a long integer value.
void exit(int val) – Terminates the program returning the value ‘val’.
EXIT_FAILURE – constant highlighting the unsuccessful termination of the program with exit(); non zero value.
EXIT_SUCCESS - constant highlighting the successful termination of the program with exit(); zero value.

#include <string.h>

char *strcpy(char *s1, char *s2) - Copies s2 in s1. Returns s1
char *strncpy(char *s1, char *s2, size_t n) – Copies the first "n" characters of s2 in s1. Returns s1.
int strcmp(char *s1, char *s2) - Compares s1 and s2 to determine the alphabetical order (<0, s1 precedes s2, 0 equal, >0 s1 follows s2)
int strncmp(char *s1, char *s2, size_t n) – Compares the first "n" characters of two strings.
int strlen(char *string) – Determines the length of a string.
char *strcat(char *s1, char *s2, size_t n) - Links s2 to s1. Returns s1
char *strncat(char *s1, char *s2, size_t n) - Links "n" characters of s2 to s1. Returns s1
char *strchr(char *string, int c) – Finds the first occurrence of the character ‘c’ in string; returns a pointer to the first occurrence of c in s, NULL if not present.
char *strrchr(char *string, int c) – Finds the last occurrence of the character ‘c’ in string.
char* strstr(char* s, char* t) – Returns a pointer to the first occurrence of t in s. returns NULL if not present.
char* strtok(char* s, const char* t) – Decomposes s in tokens, the characters that limit the tokens are contained in t. returns the pointer to the token (NULL if any is found). At the first call the string to be decomposed is s and the characters delimiting the various tokens in t. To operate on the same string, at following calls NULL has to be passed instead of s.

#include <ctype.h>

int isalnum(int c) – True if ‘c’ is alphanumeric.
int isalpha(int c) – True if ‘c’ is an alphabetic charater.
int iscntrl(int c) – True if ‘c’ is a control character.
int isdigit(int c) - True if ‘c’ is a decimal digit.
int islower(int c) - True if ‘c’ is lowercase.
int isprint(int c) - True if ‘c’ is a printable character.
int ispunct (int c) - True if ‘c’ is a punctuation character.
int isspace(int c) - True if ‘c’ is a space character.
int isupper(int c) - True if ‘c’ is uppercase.
tolower(int c) – Converts ‘c’ to lowercase.
int toupper(int c) – Convert ‘c’ to uppercase.

#include <math.h>

int abs (int n) – integer absolute value
long labs(long n) – long absolute value
double fabs (double x) – absolute value of x
double acos(double x) - arccosine
double asin(double x) - arcsin
double atan(double x) - artangent
double atan2(double y, double x) – arctangent of y/x.
double ceil(double x) – round up value of x.
double floor(double x) – round down value of x.
double cos(double x) – cos (x in radians)
double sin(double x) – sin (x in radians)
double tan(double x) – tan (x in radians)
double cosh(double x) – hyperbolic cosine
double sinh(double x) – hyperbolic sin
double tanh(double x) – hyperbolic tangent
double exp(double x) - e^x
double log(double x) - log(x).
double log10 (double x) – logarithm base 10
double pow (double x, double y) - x^y
int rand (void) – random integer between 0 and RND_MAX.
int random(int max_num) – random value between 0 and max_num.
void srand(unsigned seed) – initialize the sequence of random values
double sqrt(double x) – square root

#include <limits.h>

INT_MAX – Maximum value that can be represented by int variable.
INT_MIN – Minimum value that can be represented by int variable.
LONG_MAX - Maximum value that can be represented by long variable.
LONG_MIN - Minimum value that can be represented by long variable.

#include <float.h>

FLT_MAX, DBL_MAX - Maximum value that can be represented by float (or double) variable.
FLT_MIN, DBL_MIN - Minimum value that can be represented by float (or double) variable.

Computer Science – 05/07/2013 – **Duration: 2h** **Turn 2– Version. C**

Name and Surname	
Student ID	

Course
1(AAAA-BARA) ☐2 (BARB – BOTS) ☐3 (BOTT – CAR) ☐4 (CAS – CORD) ☐5 (CORE – DIF) ☐6 (DIG – FIOR) ☐7 (FIOS - GIORD)☐8 (GIORE – LANE) ☐9 (LANF – MARA) ☐10 (MARB – MOH) ☐11 (MOI – PAK) ☐12 (PAL – POLH) ☐13 (POLI – ROSA) ☐14 (ROSB–SIL) ☐15 (SIM – TR)☐16 (TS –ZZ) ☐E1 (AA – LZ) ☐E2 (MA – ZZ) ☐Poli@Home☐Es. (5 crediti)☐

Theory

Question 1

Convert the following numbers: 23 **from** base 10 **to** base 2, pure binary on 8 bits 100011 **from** base 2, 2's complement on 6 bits **to** base 10 0110100 **from** base 2 **to** base 16	Result
The most significant passages to arrive the result	

Question 2

Given the following two Boolean functions *f* and *g*, determine whether they are equivalent by constructing the truth tables: $f = \overline{(\bar{x} + \bar{y})z}$ $g = x\,y + \bar{z}$	Result
The most significant passages to arrive at the result	

Question 3

Briefly describe the internal structure of a CPU.

Programming

A text file contains the information relating to the presences in the Italian football main league (serie A). Write a C program to extract statistics from a sample of filtered data according to specific parameters.

The name of the file that contains the information on the matches is passed in the first argument of the command line. Each row of the file corresponds to a specific football player with the format as follows:

```
<SURNAME><NAME><#PRESENCES><MEDIA-VOTE><TEAM>
```

where SURNAME and NAME are two distinct fields where each contains no more than 40 characters (without spaces), #PRESENCES is the number of times that the player is present in an official match, MEDIA-VOTE is the average rating assigned to the player during his attendance, TEAM (with the length no more than 25 characters) is the name of the corresponding team that the player currently plays in. We also make the following assumptions:

- All the fields are separated from each other by one single space
- The number of the players is unknown
- There are no homonyms
- The content of the file is correct with respect to the format

The rules for filtering the data are contained in a second file with the name as *filter.txt*. In this file, each line lists the player names that must be excluded:

```
<TEAM><#PRESENCES>
```

The maximum number of rules present in this file is 100.
According to the filter, the program must print out:

- The list of players (with the related information) whose team is contained in *filter.txt* and the number of his presences is larger than the specific value
- The player with the lowest number of presences (among the printed names)
- The player with the largest number of presences (among the printed names)

In case of ambiguity, print the first player that is found.

Example:

serieA.txt

```
Chiellini Giorgio 24 5.32 JUVENTUS
Consigli Andrea 35 6.13 ATALANTA
Marek Hamsik 38 6.45 NAPOLI
Diamanti Alessandro 10 6.88 NAPOLI
Vucinic Mirko 5 4.3 ATALANTA
Basta Dusan 28 6.25 UDINESE
Perrotta Simone 14 5.7 ROMA
Giovinco Sebastiano 3 4.5 UDINESE
```

filter.txt

```
ATLANTA 10
MILAN 15
UDINESE 25
NAPOLI 20
INTER 3
```

```
C:\>prog.exe serieA.txt
Consigli Andrea 35 6.13 ATALANTA
Marek Hamsik 38 6.45 NAPOLI
Basta Dusan 28 6.25 UDINESE

Presence min.: 28 Basta Dusan
Presence max.: 38 Marek Hamsik
```

#include <stdio.h>

FILE *fopen(char *filename, char * mode) – Opening a file (mode: “r” reading – “w” writing – “a” append)
FILE *freopen(char *filename, char * mode, FILE *file_pointer) – Reassign a file pointer to a different file
int fclose(FILE *file_pointer) – Closing a file
int feof(FILE *file_pointer) – Checks if end-of-file has been reached.
int fflush(FILE *file_pointer) - Empties file’s buffer.
int getchar(void) – Reads a character from “stdin" (keyboard)
int fgetc(FILE *file_pointer) – Gets a character from a file
char *gets(char *buffer) - Reads a line from “stdin" (keyboard)
char *fgets(char *string, int maxchar, FILE *file_pointer) - Reads a line from a file
int printf(char *format _string, …) – Writes formatted output on "stdout" (screen)
int fprintf(FILE *file_pointer, char *format_string, …) – Writes formatted output on a file.
int sprintf(char *string, char *format_string, …) – Writes formatted output on a string.
int fputc(int c, FILE *file_pointer) – Writes a character on a file.
int putchar(int c) – Writes a character on “stdout” (screen).
int puts(char *string) - Writes a string on “stdout” (screen).
int fputs(char *string, FILE *file_pointer) - Writes a string on a file.
int scanf(char *format_string, args) – Reads formatted input from “stdin” (keyboard)
int fscanf(FILE *file_pointer, char *format string, args) – Reads formatted input from a file.
int sscanf(char *buffer, char *format_string, args) – Reads formatted input from a string.
EOF – end of file (costant with negative value)
NULL - null pointer (value 0)

#include <stdlib.h>

double atof(char *string) – Converts a string into a floating point value.
int atoi(char *string) – Converts a string into an integer value.
int atol(char *string) – Converts a string into a long integer value.
void exit(int val) – Terminates the program returning the value ‘val’.
EXIT_FAILURE – constant highlighting the unsuccessful termination of the program with exit(); non zero value.
EXIT_SUCCESS - constant highlighting the successful termination of the program with exit(); zero value.

#include <string.h>

char *strcpy(char *s1, char *s2) - Copies s2 in s1. Returns s1
char *strncpy(char *s1, char *s2, size_t n) – Copies the first "n" characters of s2 in s1. Returns s1.
int strcmp(char *s1, char *s2) - Compares s1 and s2 to determine the alphabetical order (<0, s1 precedes s2, 0 equal, >0 s1 follows s2)
int strncmp(char *s1, char *s2, size_t n) – Compares the first "n" characters of two strings.
int strlen(char *string) – Determines the length of a string.
char *strcat(char *s1, char *s2, size_t n) - Links s2 to s1. Returns s1
char *strncat(char *s1, char *s2, size_t n) - Links "n" characters of s2 to s1. Returns s1
char *strchr(char *string, int c) – Finds the first occurrence of the character ‘c’ in string; returns a pointer to the first occurrence of c in s, NULL if not present.
char *strrchr(char *string, int c) – Finds the last occurrence of the character ‘c’ in string.
char* strstr(char* s, char* t) – Returns a pointer to the first occurrence of t in s. returns NULL if not present.
char* strtok(char* s, const char* t) – Decomposes s in tokens, the characters that limit the tokens are contained in t. returns the pointer to the token (NULL if any is found). At the first call the string to be decomposed is s and the characters delimiting the various tokens in t. To operate on the same string, at following calls NULL has to be passed instead of s.

#include <ctype.h>

int isalnum(int c) – True if ‘c’ is alphanumeric.
int isalpha(int c) – True if ‘c’ is an alphabetic charater.
int iscntrl(int c) – True if ‘c’ is a control character.
int isdigit(int c) - True if ‘c’ is a decimal digit.
int islower(int c) - True if ‘c’ is lowercase.
int isprint(int c) - True if ‘c’ is a printable character.
int ispunct (int c) - True if ‘c’ is a punctuation character.
int isspace(int c) - True if ‘c’ is a space character.
int isupper(int c) - True if ‘c’ is uppercase.
tolower(int c) – Converts ‘c’ to lowercase.
int toupper(int c) – Convert ‘c’ to uppercase.

#include <math.h>

int abs (int n) – integer absolute value
long labs(long n) – long absolute value
double fabs (double x) – absolute value of x
double acos(double x) - arccosine
double asin(double x) - arcsin
double atan(double x) - artangent
double atan2(double y, double x) – arctangent of y/x.
double ceil(double x) – round up value of x.
double floor(double x) – round down value of x.
double cos(double x) – cos (x in radians)
double sin(double x) – sin (x in radians)
double tan(double x) – tan (x in radians)
double cosh(double x) – hyperbolic cosine
double sinh(double x) – hyperbolic sin
double tanh(double x) – hyperbolic tangent
double exp(double x) - e^x
double log(double x) - log(x).
double log10 (double x) – logarithm base 10
double pow (double x, double y) - x^y
int rand (void) – random integer between 0 and RND_MAX.
int random(int max_num) – random value between 0 and max_num.
void srand(unsigned seed) – initialize the sequence of random values
double sqrt(double x) – square root

#include <limits.h>

INT_MAX – Maximum value that can be represented by int variable.
INT_MIN – Minimum value that can be represented by int variable.
LONG_MAX - Maximum value that can be represented by long variable.
LONG_MIN - Minimum value that can be represented by long variable.

#include <float.h>

FLT_MAX, DBL_MAX - Maximum value that can be represented by float (or double) variable.
FLT_MIN, DBL_MIN - Minimum value that can be represented by float (or double) variable.